SQUINT Fields, Maps, Patterns, and Lattices

Jarek Rossignac

GVU Center Technical Report GVU-2018-04
http://hdl.handle.net/1853/60061
School of Interactive Computing, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

July 23, 2018

Abstract

The proposed Steady QUad INTerpolating (SQUINT) map is formulated in terms of a SQUINT Field of Similarities (FoS). It is controlled by four coplanar points. It maps the unit square onto a curved planar quad, **R**, which has these points as corners. Uniformly spaced, log-spiral isocurves decompose **R** into tiles that are similar to each other and, hence, each have equal angles at opposite corners. We provide closed-form expressions for computing the representation of the SQUINT map and for evaluating the map and its inverse. We discuss extensions and potential applications to texture maps and field warps and to the design, display, and constant-cost query of procedural models of arbitrarily complex lattices.

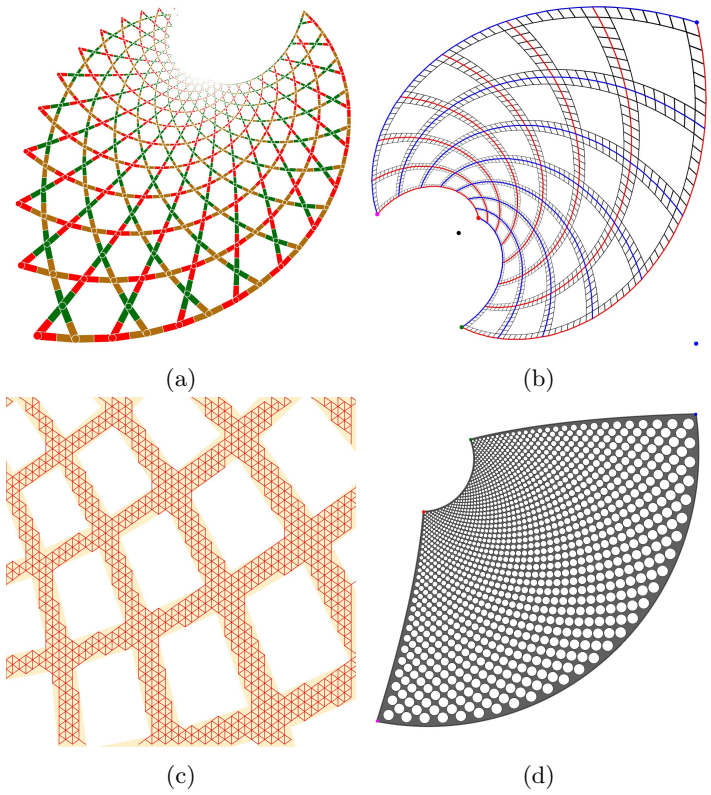


Figure 1: A *SQUINT* lattice with Kagome connectivity (a). A procedurally defined sparse multi-scale version of a *SQUINT* lattice (b). A detail of a LiL (c). A material structure defined by a *SQUINT* field of holes with varying radius that has a constant homogenized mass (d).

1 INTRODUCTION

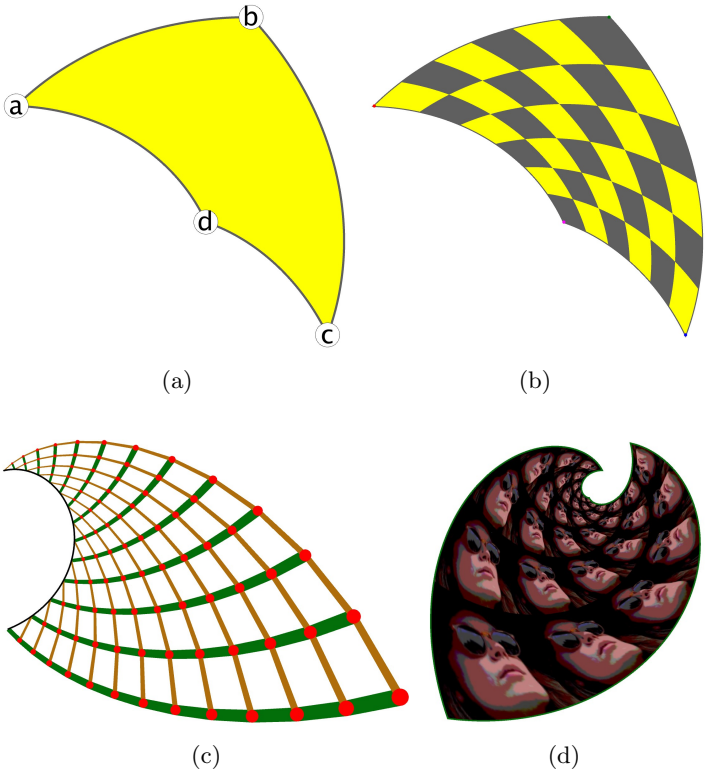
1.1 Motivation

The overarching motivation for this research is to simplify the design of procedural models of extremely complex lattice structures [17, 19, 3, 22] and to accelerate important query on them. We explore planar *SQUINT* lattices (see Fig. 1-a for an example). We exploit their steadiness to accelerate geometric queries, such as Point-Membership Classification [7] and computation of Integral Properties (surface, volume, or mass) [4] and to support procedurally trimmed variations (Fig. 1-b) and procedural multi-scale Lattice-in-Lattice (LiL) models [10, 7], where a coarse lattice is used as a solid-trim for a finer one (Fig. 1-c). We also show their applicability to control simple scalar and vector fields, which may dictate orientation and density of fibers, or the spatial gradation of features for which we can easily compute effective material properties (Fig. 1-d).

1.2 Key contributions

The ***Steady Quad Interpolating (SQUINT)*** paradigm proposed in this paper makes it possible to meet the following goals: ***(1)*** Provide an easy-to-use and effective interactive tool for designing or animating a simple warp of a square domain that, in a sense that we clarify in this paper, distributes the uniform scaling steadily and the residual deformation evenly. ***(2)*** Support the design and possibly beautification ([8]) of geometric two-patterns that have a coherent and pleasing variation in element position, orientation, and scale along a curved coordinate system. The detection of patterns in images has been investigated extensively by the Computer Vision community and more recently by the Geometry Processing community (see for example [11, 13]). The robust detection of steady geometric patterns

was addressed in [16, 9]. ***(3)*** Use the above to accelerate the processing of procedural models of steady two-directional lattices [4]. *SQUINT* maps the square, $\mathbf{D} = [0,1]^2$, of the parametric domain, to a planar image-shape, $\mathbf{R}$, that is bounded by four log-spiral segments. *SQUINT* is fully controlled by the locations of the four ***control corners*** ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$) of $\mathbf{R}$ (Fig. 2-a). It maps (Fig. 2-b) the square cells of an $n \times n$ regular grid partition of $\mathbf{D}$ onto a self-similar two-pattern of tiles, $\mathbf{T}_{i,j}$, for which opposite angles are identical. We provide simple and exact, closed-form expressions for computing the *SQUINT* map and its inverse and for identifying, in constant time (regardless of the value of n), the tile that contains a query point. Finally, we conjecture that *SQUINT* lattices may offer a useful generalization of some variants of Michell trusses [21, 23], which minimize weight for a given load (Fig. 2-c) and that it may be useful for creating artistic arrangements of steady patterns (Fig. 2-d).



(a) (b) (c) (d)

Figure 2: The range, $\mathbf{R}$, of a *SQUINT* map and its four control corners (a), its tiles, which are similar to each other (b), the green tile (c) a steady variation of Michell trusses, in which all tiles are similar (d), and an steady pattern of an image.

1.3 Organization of the paper

The remainder of the paper is organized as follows. In Section 2, we define steady fields (two-parameter families) of planar similarities that are controlled by four corners points. In Section 3, we define a planar map in terms of such a *SQUINT* field, present the closed-form of its inverse, and discuss its isocurves, tiles, and important properties. In Section 4, we discuss the properties of steady two-patterns of shapes defined by a *SQUINT* field and a fast (constant cost) pick-tile query, which is based on the *SQUINT* map inversion. In Section 5, we define a planar disk-and-bar lattice in terms of *SQUINT* patterns of disks and show that, when it is the assembly of quasi-disjoint hubs [4], the hubs form one or more *SQUINT* patterns and hence can be queried, in constant time-cost, for point-membership and for some integral properties.

2 *SQUINT* FIELD

In this section, we first define our notation and review basic facts on planar similarities. Then, we define curves and fields of similarities, their steady version, and then the *SQUINT* field, which is steady and controlled by four points.

2.1 Notation

We use the following notation in our geometric constructions. Important technical terms are written in bold when they are introduced or defined. ***Integers*** are written as lowercase letters in normal font, such as i, j and n. ***Scalars***, which are used to denote mapping parameters, scaling ratios, or angles, are written in lowercase italics, such as u, v, t, λ, or α. ***Points*** are denoted by lowercase, bold letters, such as $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$, $\mathbf{f}$, $\mathbf{p}$, or $\mathbf{p}_i$ or in terms of their Cartesian coordinates using parentheses, such as (x, y). ***Vectors*** are denoted by lowercase, bold letters with an overhead arrow, such as $\vec{\mathbf{u}}$ or $\vec{\mathbf{v}}$, or in terms of their Cartesian components using brackets, such as $\langle x, y \rangle$. Vector from point $\mathbf{a}$ to point $\mathbf{b}$ is written $\overrightarrow{\mathbf{ab}}$. Its ***magnitude***, i.e., the distance from $\mathbf{a}$ to $\mathbf{b}$ is denoted $|\mathbf{ab}|$. Addition $\mathbf{p} + \vec{\mathbf{v}}$ defines a point obtained by translating $\mathbf{p}$ by vector $\vec{\mathbf{v}}$. The ***angle*** between vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ is denoted $\vec{\mathbf{u}}^{\wedge}\vec{\mathbf{v}}$. The ***dot product*** between them is denoted $\vec{\mathbf{u}} \bullet \vec{\mathbf{v}}$. The version of $\vec{\mathbf{v}}$ ***rotated*** by angle α is written $\vec{\mathbf{v}}^{\circ}\alpha$. Hence, $\overrightarrow{\mathbf{pf}}^{\circ}(t\alpha)$ denotes the result of rotating vector $\overrightarrow{\mathbf{pf}}$ by angle $t\alpha$. Shortcut $\vec{\mathbf{u}}$ stands for $\vec{\mathbf{u}}^{\circ}(\pi/2)$. Hence, $\vec{\mathbf{u}}^{\bullet\vec{\mathbf{v}}}$ measures the dot product $(\vec{\mathbf{u}}) \bullet \vec{\mathbf{v}}$ between $\vec{\mathbf{v}}$ and a version of $\vec{\mathbf{u}}$ rotated by $\pi/2$. ***Shapes*** (i.e., continuous pointsets), such a (curved) quad, disk or edge, are written using uppercase bold letters, such as $\mathbf{D}$, $\mathbf{X}$, or $\mathbf{T}_{i,j}$. ***Transformations***, for example similarities, are written using curly uppercase letters, such as $\mathcal{S}$, $\mathcal{U}$, or $\mathcal{V}$. ***Composition*** (i.e., product) of transformations $\mathcal{U}$ and $\mathcal{V}$ is denoted by $\mathcal{U} \cdot \mathcal{V}$. The result of ***applying transformation*** $\mathcal{U}$ to point $\mathbf{p}$ is denoted $\mathcal{U} \cdot \mathbf{p}$. The result of applying it to shape $\mathbf{X}$ is denoted $\mathcal{U} \cdot \mathbf{X}$. Cascaded product $\mathcal{U} \cdot \mathcal{V} \cdot \mathcal{W} \cdot \mathbf{a}$ means $\mathcal{U} \cdot (\mathcal{V} \cdot (\mathcal{W} \cdot \mathbf{a}))$. Notation $\mathcal{U}^t$ refers to the scalar ***power of transformation*** $\mathcal{U}$. For example, $\mathcal{U}^3 = \mathcal{U} \cdot \mathcal{U} \cdot \mathcal{U}$ and, if $\mathcal{U} = \mathcal{S}^{1/2}$, then $\mathcal{U}^2 = \mathcal{S}$. The ***oriented edge*** (closed line segment) from $\mathbf{a}$ to $\mathbf{b}$ is written $\mathbf{ab}$. The ***labelled triangle*** with vertices labelled $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ is written $\mathbf{abc}$. Finally, $\mathbf{abc} \sim \mathbf{def}$ indicates that these triangles are ***similar*** (i.e., related by a similarity).

2.2 Similarities

In this paper, we mostly restrict our scope to pointsets, arrangements, transformations, and mappings in the plane. We formulate our solutions in terms of affinities (affine transformations) [14]. They are invertible and preserve collinearity, i.e., map straight edges to straight edges.

We use the following set of ***primitive*** affinities. ***Translation***, $\mathcal{T}_{\vec{\mathbf{v}}}$, by vector $\vec{\mathbf{v}}$ maps point $\mathbf{p}$ to point $\mathcal{T}_{\vec{\mathbf{v}}} \cdot \mathbf{p} = \mathbf{p} + \vec{\mathbf{v}}$. ***Rotation***, $\mathcal{R}_{\mathbf{f},\alpha}$, by angle α around ***center*** (i.e., fixed point) $\mathbf{f}$ maps $\mathbf{p}$ to $\mathcal{R}_{\mathbf{f},\alpha} \cdot \mathbf{p} = \mathbf{f} + \overrightarrow{\mathbf{fp}}^{\circ} \alpha$. ***Dilation***, $\mathcal{D}_{\mathbf{f},\lambda}$, by a uniform scaling ratio λ about center $\mathbf{f}$ maps $\mathbf{p}$ to $\mathcal{D}_{\mathbf{f},\lambda} \cdot \mathbf{p} = \mathbf{f} + \lambda \overrightarrow{\mathbf{fp}}$.

A ***similarity*** is any combination of these primitive transformations. It preserves angles (i.e., it is ***conformal***) and collinearity. A ***true similarity*** is a similarity that is not a pure primitive translation. A true similarity, $\mathcal{S}$, has a unique ***center*** (i.e., ***fixed point***), $\mathbf{f}$, and a unique ***canonical decomposition*** into the product $\mathcal{R}_{\mathbf{f},\alpha} \cdot \mathcal{D}_{\mathbf{f},\lambda}$ of dilation $\mathcal{D}_{\mathbf{f},\lambda}$ by ratio λ with respect to $\mathbf{f}$ and rotation $\mathcal{R}_{\mathbf{f},\alpha}$ by angle α around $\mathbf{f}$. Hence, $\mathcal{S}$ can be ***represented*** by the triplet $< \mathbf{f}, \lambda, \alpha >$ of the values of its ***canonical parameters***: ***center*** $\mathbf{f}$, ***scaling*** factor λ, and rotation ***angle*** α. We evaluate the image, $\mathcal{S} \cdot \mathbf{p}$, of point $\mathbf{p}$ by similarity $\mathcal{S} =< \mathbf{f}, \lambda, \alpha >$ using $\mathcal{D}_{\mathbf{f},\lambda} \cdot (\mathcal{R}_{\mathbf{f},\alpha} \cdot \mathbf{p})$, which equals $\mathbf{f} + \lambda \overrightarrow{\mathbf{fp}}^{\circ} \alpha$.

Shapes $\mathbf{X}$ and $\mathbf{Y}$ are ***similar*** (i.e., $\mathbf{X} \sim \mathbf{Y}$), when there exists a similarity, $\mathcal{S}$, such that $\mathbf{Y} = \mathcal{S} \cdot \mathbf{X}$. Observe that two oriented edges are always similar.

Consider four different points, $\mathbf{a}_0$, $\mathbf{b}_0$, $\mathbf{a}_1$, and $\mathbf{b}_1$ and let $\mathcal{S} = \mathrm{Sim}(\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1)$ define $\mathcal{S}$ to be the similarity satisfying $\mathbf{a}_1 = \mathcal{S} \cdot \mathbf{a}_0$ and $\mathbf{b}_1 = \mathcal{S} \cdot \mathbf{b}_0$. Because similarities preserve collinearity, we may equivalently say that $\mathcal{S}$ maps oriented edge $\mathbf{a}_0\mathbf{b}_0$ to oriented edge $\mathbf{a}_1\mathbf{b}_1$. $\mathrm{Sim}(\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1)$ computes and returns the values of $< \mathbf{f}, \lambda, \alpha >$ as follows: $\lambda = |\mathbf{a}_1\mathbf{b}_1| \div |\mathbf{a}_0\mathbf{b}_0|$, $\alpha = \overrightarrow{\mathbf{a}_0\mathbf{b}_0} \wedge \overrightarrow{\mathbf{a}_1\mathbf{b}_1}$, and $\mathbf{f}$ is the solution of the linear system of two equations defined by vector equality $\overrightarrow{\mathbf{fa}_1} = \lambda \overrightarrow{\mathbf{fa}_0}^{\circ} \alpha$. We solve it using $\mathbf{f} = \mathbf{a}_0 + \langle \vec{\mathbf{w}} \bullet \overrightarrow{\mathbf{a}_1\mathbf{a}_0}, \vec{\mathbf{w}} \bullet \overrightarrow{\mathbf{a}_1\mathbf{a}_0}^{\perp} \rangle \div d$, with vector $\vec{\mathbf{w}} = \langle \lambda \cos\alpha - 1, \lambda \sin\alpha \rangle$. The determinant d of this system is $\vec{\mathbf{w}}^2 = (\lambda \cos\alpha - 1)^2 + (\lambda \sin\alpha)^2$. It can be null only when both $\sin\alpha = 0$ (parallelism of opposite edges) and $\lambda = 1$ (no scaling), thus implying a primitive translation. Hence, $\mathrm{Sim}(\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1)$ returns a true similarity when $\overrightarrow{\mathbf{a}_0\mathbf{b}_0} \neq \overrightarrow{\mathbf{a}_1\mathbf{b}_1}$, i.e., when the control polygon is not a parallelogram.

2.3 Curves of Similarities

Let us start by defining the steady interpolation (morph) between two vectors, $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$. We use the ***Log-Polar Morph*** (***LPM***), which we define to yield vector $\vec{\mathbf{w}}(t) = \lambda^t \, \vec{\mathbf{u}}^{\circ}(t\alpha)$, with $\lambda = |\vec{\mathbf{v}}| \div |\vec{\mathbf{u}}|$ and $\alpha = \vec{\mathbf{u}} \wedge \vec{\mathbf{v}}$. This vector-motion is ***steady*** [18], meaning that $\vec{\mathbf{w}}(t+u)$, which can be written $\lambda^u \lambda^t \, \vec{\mathbf{u}}^{\circ}(u\alpha + t\alpha)$, or equivalently $\lambda^u(\vec{\mathbf{w}}(t))^{\circ}(u\alpha)$, may be obtained by applying a transformation to $\vec{\mathbf{w}}(t)$ that depends on u, but not on t. This steadiness implies that the wedges between the consecutive vectors, $\vec{\mathbf{w}}(\mathrm{i}/(\mathrm{n}\text{-}1))$, as shown in (Fig. 3-a), are similar.

Observe that LPM interpolates orientations linearly and magnitudes exponentially. It is compared to LERP (Fig. 3-b), which uses linear interpolation of the Cartesian coordinates, to SLERP [20] (Fig. 3-c), which uses $\vec{\mathbf{w}}(t) = \langle \sin((1-t)\alpha), \sin(t\alpha) \rangle \div \sin\alpha$ and is equivalent to LPM (and only useful) when $|\vec{\mathbf{v}}| = |\vec{\mathbf{u}}|$, and to Polar Morph, which interpolates linearly both orientation and length and resembles LPM. In general, LERP, SLERP, Polar Morph are not steady.

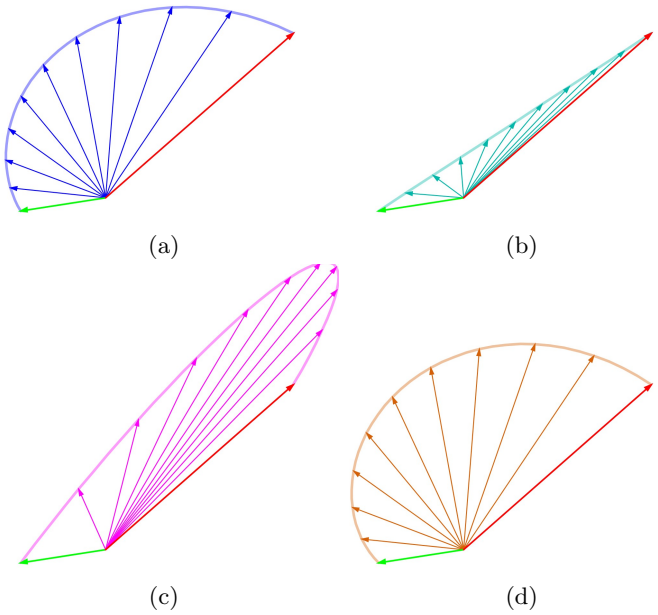


Figure 3: Interpolation between two vectors: Log-Polar Morph (LPM), which is steady (a), LERP (b), SLERP (c), and Polar Morph (d), which differs from LPM because it uses a linear, rather than exponential, interpolation of the magnitude and hence is not steady.

Instead of defining the motion $\mathbf{p}(t)$ of a point, let us define the time-parameterized behavior of a similarity (i.e., the motion of the whole plane). Specifically, we define a ***Curve of Similarities*** (***CoS***) to be a continuous, one-parameter family, $\mathcal{S}_t$, of similarities. For every value of parameter t, $\mathcal{S}_t$ is a similarity. Given a template shape, $\mathbf{X}_0$, when parameter t represents time, $\mathbf{X}(t) = \mathcal{S}_t \cdot \mathbf{X}_0$ defines an animation that continuously translates, rotates, and scales $\mathbf{X}_0$.

A ***Steady Curve of Similarities*** (***SCS***) is a Curve of Similarities for which there exists a similarity $\mathcal{U}$, such that $\mathcal{S}_t = \mathcal{U}^t$. When t represents time, a SCS is a ***Steady Similarity Motion***, which is a version of the ***Steady Affine Motion*** (SAM), discussed in [18], but for which only uniform scaling is permitted. The instantaneous position, $\mathbf{p}(t)$, of a point as it is moved by such an SCS, with $\mathcal{S} =< \mathbf{f}, \lambda, \alpha >$, may be computed as $\mathbf{p}(t) = \mathbf{f} + \lambda^t \overrightarrow{\mathbf{fp}_0}^{\circ}(t\alpha)$, given its initial position, $\mathbf{p}_0 = \mathbf{p}(0)$. This point traces a logarithmic spiral. With this steady parameterization, the angle varies with constant angular velocity (as $t\alpha$) and the distance from $\mathbf{f}$ varies exponentially (as λ^t).

2.4 Fields of Similarities

Adding a second parameter, we define a Field of Similarities to be a continuous, two-parameter family, $\mathcal{S}_{u,v}$, of similarities. For every value-pair (u, v) of parameters, $\mathcal{S}_{u,v}$ is a similarity. We assume that, except at singular configurations, its derivatives with respect to these parameters are continuous.

$\mathcal{S}_{u,v}$ is a ***Regular Field of Similarities*** (***RFS***) when there exist two vectors, $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$, such that $\mathcal{S}_{u,v}$ is a translation by vector $u\,\vec{\mathbf{u}} + v\,\vec{\mathbf{v}}$.

$\mathcal{S}_{u,v}$ is a ***Product of Steady Curves of Similarities*** (***PSCS***) when there exists a pair of true similarities, $\mathcal{U}$ and $\mathcal{V}$, such that $\mathcal{S}_{u,v} = \mathcal{V}^v \cdot \mathcal{U}^u$. In that case, $\mathcal{S}_{u,v}$ is an SCS of the SCS $\mathcal{U}^u$. A generalization of PSCS to 3D and to affinities was presented in [18]. A generalization of PSCS to 3D similarities was used [7, 4] for defining lattices and for accelerating queries on them. We define next a special case of PSCS, which we qualify as steady. Its unique property is the corner stone of the *SQUINT* concepts (field, map, pattern, and lattice) proposed here.

$\mathcal{S}_{u,v}$ is a ***Steady Field of Similarities*** (abbreviated ***SFS***) when there exists a pair of true similarities, $\mathcal{U}$ and $\mathcal{V}$, such that $\mathcal{S}_{u,v} = \mathcal{V}^v \cdot \mathcal{U}^u$ and $\mathcal{V} \cdot \mathcal{U} = \mathcal{U} \cdot \mathcal{V}$.

Hence, $\mathcal{S}_{u,v}$ is a an SFS (or equivalently a ***Commutative PSCS***) when it is a PSCS for which $\mathcal{U}$ and $\mathcal{V}$ commute. This additional constraint guarantees useful properties (the key one is presented next, others are discussed further in the paper) and allows us to reduce the computational cost of important queries on patterns and lattices.

Consider the ***control quad***, $\mathbf{P}$, defined by joining the ordered ***control corners***, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$. $\mathbf{P}$ is ***non-singular***, when its vertices are all different and when it is not a parallelogram. From now on, unless specified otherwise, we will assume that $\mathbf{P}$ is non-singular. Assume that $\mathcal{U}$ maps oriented edge $\mathbf{ad}$ to oriented edge $\mathbf{bc}$ (i.e, $\mathcal{U} = \mathrm{Sim}(\mathbf{a}, \mathbf{d}, \mathbf{b}, \mathbf{c})$) and that $\mathcal{V}$ maps $\mathbf{ab}$ to $\mathbf{dc}$ (i.e, $\mathcal{V} = \mathrm{Sim}(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{c})$). The key property is that $\mathcal{U}$ and $\mathcal{V}$ have the same center $\mathbf{f}$, which we call the ***similarity center*** of the quad with vertices $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$. This surprising and important discovery leads to the following, more general, "Quad Similarities" theorem.

$\mathbf{fad} \sim \mathbf{fbc} \iff \mathbf{fab} \sim \mathbf{fdc}$. Let us prove the following, equivalent version of this theorem: If $\mathcal{U} = \mathrm{Sim}(\mathbf{a}, \mathbf{d}, \mathbf{b}, \mathbf{c}) = < \mathbf{f}_u, \lambda_u, \alpha_u >$ and $\mathcal{V} = \mathrm{Sim}(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{c}) =< \mathbf{f}_v, \lambda_v, \alpha_v >$, then $\mathbf{f}_u = \mathbf{f}_v$. Consider the images of corners $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ in the log-polar coordinate system centered at $\mathbf{f}_u$. (In log-polar coordinates, the abscissa measures the log of uniform scaling and the ordinate measures rotation angle.) They form the vertices of a parallelogram, because the log-polar versions of vectors $\overrightarrow{\mathbf{ab}}$ and $\overrightarrow{\mathbf{dc}}$ are identical. Indeed, they both have log-polar coordinates $< \log \lambda_u, \alpha_u >$, because $\mathbf{b} = \mathcal{U} \cdot \mathbf{a}$ and $\mathbf{c} = \mathcal{U} \cdot \mathbf{d}$ (Fig. 4-a). Hence, in that log-polar coordinate system, vectors $\overrightarrow{\mathbf{ad}}$ and $\overrightarrow{\mathbf{bc}}$ are also identical (since they correspond to opposite sides of that parallelogram). Consequently, $\mathbf{f}_u\mathbf{ad} \sim \mathbf{f}_u\mathbf{bc}$ (i.e., triangles $\mathbf{f}_u\mathbf{ad}$ and $\mathbf{f}_u\mathbf{bc}$ are similar in the Cartesian system) (Fig. 4-b), which implies that $\mathbf{f}_u$ is also the unique fixed point of $\mathcal{V}$ and hence identical to $\mathbf{f}_v$. The theorem holds even when the quad is self-crossing (Fig. 4-c and d). From now on, we use $\mathbf{f}$ when referring to this common center. Note that two true similarities, $\mathcal{U}$ and $\mathcal{V}$, commute if and only if they have the same center. We now define a version of an SFS that is controlled by the corners of a non-singular quad.

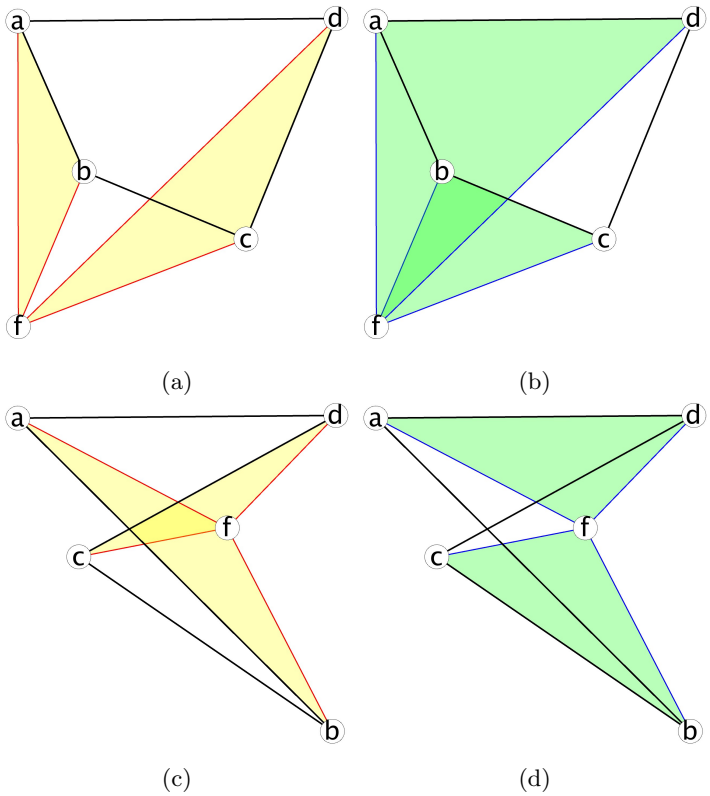


Figure 4: Given the four corners ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$) of a true (non-parallelogram) quad and its center $\mathbf{f}$ defined such that $\mathbf{fab} \sim \mathbf{fdc}$ (a), then $\mathbf{fad} \sim \mathbf{fbc}$ (b). This property holds even when the quad is self-crossing (c and d).

$\mathcal{S}_{u,v}$ is the ***SQUINT Field*** defined by four control corners, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$, when $\mathcal{S}_{u,v} = \mathcal{V}^v\,\mathcal{U}^u$, with $\mathcal{U} = \mathrm{Sim}(\mathbf{a}, \mathbf{d}, \mathbf{b}, \mathbf{c})$ and $\mathcal{V} = \mathrm{Sim}(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{c})$.

Consider the *SQUINT* field $\mathcal{S}_{u,v} = \mathcal{V}^v \cdot \mathcal{U}^u$, with $\mathcal{U} =< \mathbf{f}_u, \lambda_u, \alpha_u >$ and $\mathcal{V} =< \mathbf{f}_v, \lambda_v, \alpha_v >$. Because, by the Quad Similarity Quad Similarity Theorem, the *SQUINT* field is an SFS, the image, $\mathbf{p}(u, v) = \mathcal{S}_{u,v} \cdot \mathbf{p}$, of template point, $\mathbf{p}$, by $\mathcal{S}_{u,v}$ may now be evaluated using the simplified expression $\mathbf{p}(u, v) = \mathbf{f} + \lambda_u^u \lambda_v^v \overrightarrow{\mathbf{fp}}^{\circ}(u\alpha_u + v\alpha_v)$. We will exploit the simplicity of this expression when defining the *SQUINT* map, its inverse, and its properties (Section 3). We will also use this expression to define *SQUINT* patterns and lattices (Sections 4 and 5).

3 *SQUINT* MAP

In this section, we define the *SQUINT* map and its inverse, and examine its properties and relations to bilinear and conformal maps.

3.1 Definition

The ***SQUINT map***, $\mathbf{m}(u, v)$, associated with four distinct control corners, maps, parameter pair (u, v) in $\mathbf{D} = [0, 1]^2$ to the range-image $\mathbf{R}$. It is defined using the *SQUINT* field as follows.

The ***SQUINT Map***, $\mathbf{m}(u, v)$, defined by four control corners, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$, is $\mathcal{V}^v \cdot \mathcal{U}^u \cdot \mathbf{a}$, where $\mathcal{U} = \mathrm{Sim}(\mathbf{a}, \mathbf{d}, \mathbf{b}, \mathbf{c})$ and $\mathcal{V} = \mathrm{Sim}(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{c})$.

As explained in Sections 2, due to Quad Similarity Theorem, $\mathbf{m}(u, v)$ may be evaluated as $\mathbf{m}(u, v) = \mathbf{f} + \lambda_u^u \lambda_v^v \overrightarrow{\mathbf{fa}}^{\circ}(u\alpha_u + v\alpha_v)$.

3.2 Isocurves and tiles

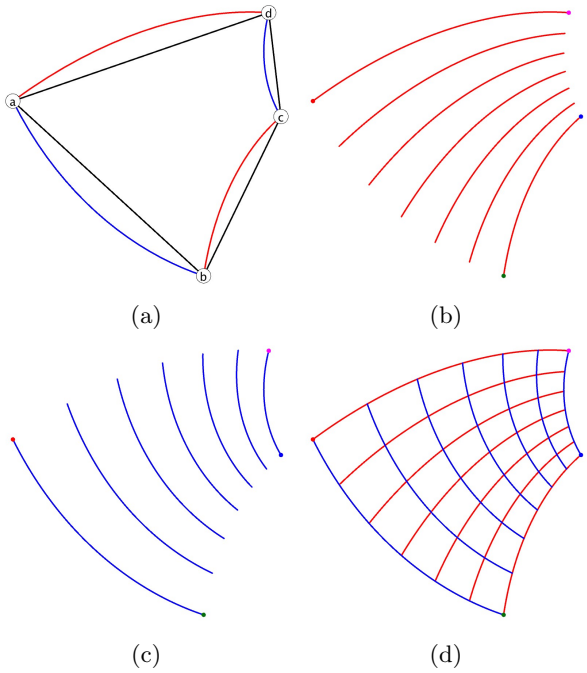


Figure 5: The *SQUINT* map border (red and blue) drawn (a) over the control polygon (black). The steady one-pattern (b) of its u-curves (red) for 7 uniformly-spaced u-values. The corresponding configuration (c) of v-curves (blue). The union of these 7 u-curves and of 7 v-curves decomposes R into a grid of 6×6 tiles (c) that forms a *SQUINT* pattern (see Section 4)

The ***isocurves*** of the *SQUINT* map for extreme values (0 or 1) of v or u are shown (Fig. 5-a). They form the border of $\mathbf{R}$ and never cross. We show them drawn over the control polygon. The term ***u-curve*** refers to an isocurve for a fixed value of u. Its parametric expression can be written $\mathbf{m}_u(v) = \mathbf{f} + \lambda_u^u \lambda_v^v \overrightarrow{\mathbf{fa}}^{\circ}(u\alpha_u + v\alpha_v)$, which, due to the commutativity of arithmetic operations, is $\mathbf{m}_u(v) = \mathbf{f} + \lambda_v^v \overrightarrow{\mathbf{fq}}^{\circ}(v\alpha_v)$, with $\mathbf{q} = \mathbf{f} + \lambda_u^u \overrightarrow{\mathbf{fa}}^{\circ}(u\alpha_u)$. This is the formulation of a log-spiral with center $\mathbf{f}$. By symmetry, every ***v-curve*** has a similar expression and is also a log-spiral. Now consider the set of n+1 u-curves (red in Fig. 5-b) for u=i/n, where integer i varies from 0 to n. Similarly, consider the set of n+1 v-curves (blue in Fig. 5-c) for v=j/n, where integer j varies from 0 to n. When superimposed (Fig. 5-d), these two families of isocurves decompose $\mathbf{R}$ into a ***two-pattern*** of n × n ***tiles***. $\mathbf{T}_{\mathrm{i,j}}$ denotes the tile bounded by segments of u-curves for u = i/n and u = (i + 1)/n and of v-curves for v = j/n and v = (j + 1)/n.

3.3 Relation to bilinear interpolation

SQUINT is built from similarities that each map an edge to another. Consequently, it inherits the associated singularity issues. Specifically, in singular configurations, where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ are the consecutive vertices of a parallelogram, *SQUINT* degenerates to a ***bilinear interpolation*** map: $\mathbf{m}(u, v) = \mathbf{i}(\mathbf{i}(\mathbf{a}, u, \mathbf{b}), v, \mathbf{i}(\mathbf{d}, u, \mathbf{c}))$, where $\mathbf{i}(\mathbf{a}, u, \mathbf{b})$ is the LERP from $\mathbf{a}$ to $\mathbf{b}$ that returns $\mathbf{a} + u\,\overrightarrow{\mathbf{ab}}$.

Observe (Fig. 6) that, as the control polygon smoothly evolves towards a parallelogram, *SQUINT* converges smoothly towards the bilinear interpolation. Such singular configurations are trivially detected ($\overrightarrow{\mathbf{ab}} = \overrightarrow{\mathbf{dc}}$), but numeric issues in the vicinity of these special configurations need to be addressed carefully and a blending between the two solutions may be desired near singular configurations. For simplicity, from now on, we assume that the control polygon is not a parallelogram.

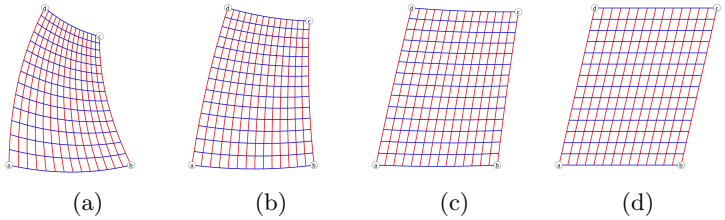


Figure 6: *SQUINT* converges smoothly towards a bilinear interpolation (d) as its control polygon converges towards a parallelogram (from left to right) by adjusting corner $\mathbf{c}$ in the up-right direction.

The mapping from the current positions of control corners ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$) to the *SQUINT* map (shown (Fig. 7 by displaying its isocurves and tiles) is ***quasi-continuous***, even if the control-polygon self-crosses. For example, starting from a control-polygon that is not self-crossing (Fig. 7-a), as control corner c is dragged downward by a continuous motion, we obtain a self-crossing configuration (Fig. 7-b). During that continuous motion of $\mathbf{c}$, the map $\mathbf{m}$ remains valid and deforms continuously. Dragging $\mathbf{c}$ further down reaches a degenerate configuration, in which corners $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$, are all on the same log-spiral curve and which yields a collapsed (one-dimensional) *SQUINT* quad $\mathbf{R}$ (Fig. 7-c). Dragging $\mathbf{c}$ further down restores a non-degenerate configuration, in which the initially-clockwise orientation of the tiles is now reversed (Fig. 7-d).

We used the term ***quasi-continuous*** above, because, in some extremely deformed configurations, the *SQUINT* map may flip. For example, consider the configuration in Fig. 8-a. Continuously dragging corner $\mathbf{a}$ slightly to the left produces a discontinuous behavior of $\mathbf{R}$, which flips (Fig. 8-b). Such flips occur when angle $\overrightarrow{\mathbf{ab}}^{\wedge}\overrightarrow{\mathbf{dc}}$ crosses the π or $-\pi$ mark, because the

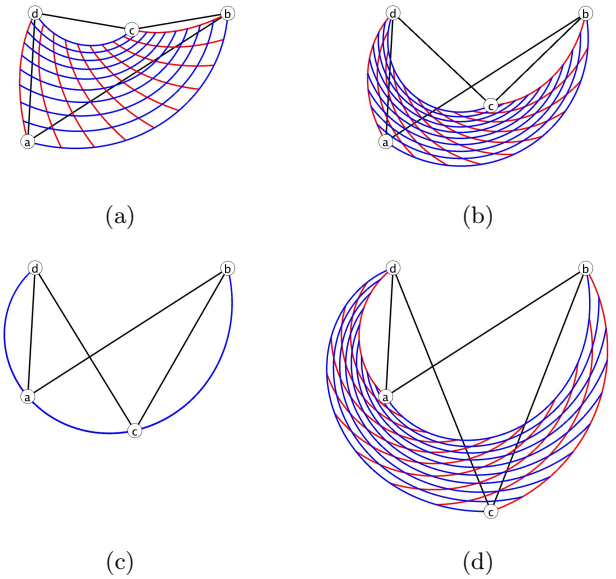


Figure 7: Starting from the gentle deformation (a), we drag control corner **c** downward. During that motion, the *SQUINT* map evolves in a continuous manner, even though the control-polygon becomes self-crossing (b). Dragging **c** further down yields a degenerate configuration, where **R** is collapsed to a log-spiral curve (c). Dragging **c** further down, restores a valid configuration (d).

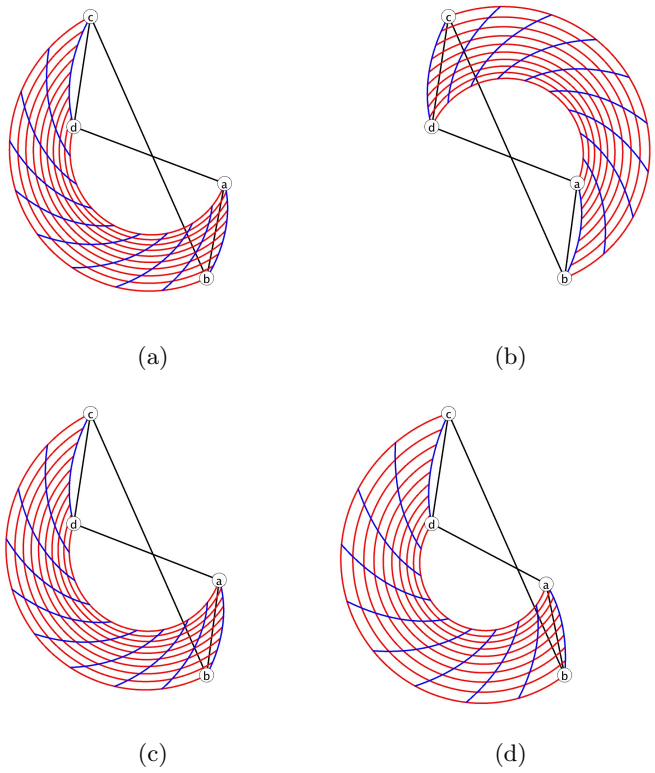


Figure 8: Tiles for a self-crossing control polygon (a). Dragging control corner **a** slightly to the right results in a flip (b). Adjusting the angle function prevents the flip (c) and results in a locally continuous behavior, whether one drags **a** back to the left (a) or forward to the right (d).

angle function that we use returns an angle β in $[-\pi, \pi]$. Replacing this function temporarily by one that returns an angle in $[0, 2\pi]$ (by returning $\beta + 2\pi$ when $\beta < 0$ fixes the problem here (Fig. 8-c) and allows the user to continue dragging **a** without the undesired flip (Fig. 8-d). Such a user-controlled switch may be acceptable in some interactive applications or for constrained optimization domains. For some applications, it may be appropriate to perform the switch automatically when the angle inversion is detected, as one would in inverse kinematics for robotics or CNC to avoid the infinite acceleration when switching between elbow-up/elbow-down configurations.

We compare (Fig. 9) the *SQUINT* map to the map defined by the Bilinear interpolation for the same corner points. Observe how the tiles of the *SQUINT* map (Fig. 9-b) appear more regular than those of the bilinear map (Fig. 9-a). We also show (Fig. 9-c) a configuration where the bilinear map folds (i.e., self-overlaps and hence does not have a unique inverse in **R**). Observe (Fig. 9-d) that *SQUINT* handles such a challenging input gracefully, provided that control corners are all different and not on the same log-spiral.

3.4 Transimilarity

We say that a map is ***tranSimilar*** when a translation by arbitrary vector $\vec{\mathbf{w}}$ in parameter space corresponds to a similarity $\mathcal{S}_{\vec{\mathbf{w}}}$ in the image space, where $\mathcal{S}_{\vec{\mathbf{w}}}$ only depends on $\vec{\mathbf{w}}$, and not on what is transformed nor where it is. To prove that the *SQUINT* map is tranSimilar, consider translation vector $\vec{\mathbf{w}} = \langle x, y \rangle$. Let $\mathbf{p} = \mathcal{V}^v \cdot \mathcal{U}^u \cdot \mathbf{a}$. A translation by $\vec{\mathbf{w}}$ in parametric space produces $\mathbf{q} = \mathcal{V}^{v+y} \cdot \mathcal{U}^{u+x} \cdot \mathbf{a}$, which can be written as $\mathbf{q} = (\mathcal{V}^y \cdot \mathcal{U}^x) \cdot (\mathcal{V}^v \cdot \mathcal{U}^u \cdot \mathbf{a})$. Hence, $\mathbf{q} = \mathcal{S}_{\vec{\mathbf{w}}} \cdot \mathbf{p}$, with $\mathcal{S}_{\vec{\mathbf{w}}} = \mathcal{V}^y \cdot \mathcal{U}^x$, a similarity independent of u and v, and hence of **p**.

The tranSimilarity of the *SQUINT* map is illustrated in Fig. 10-a, where we start with a magenta shape **X** and make a green copy **Y** of it obtained by translating the pre-image of **X** in parameter space. To demonstrate that **X** and **Y** are similar, we make a copy of the whole region **R**, which contains both **X** and **Y**. Then we apply a similarity to that copy to align it so that its green shape **Y** overlaps with the original version of the magenta shape **X**. They match perfectly. TranSimilarity is important when we want to produce seamless overlap of similarity copies of the tiling of a *SQUINT* map (Fig. 10-d) or when we wish to extend it past R and ensure a seamless covering. We say that a map is ***rotSimilar*** if an arbitrary rotation in parameter space corresponds to a similarity in the image space and that it is ***dilSimilar*** if an arbitrary dilation in parameter space corresponds to a similarity in the image space. Unfortunately, as shown in (Fig. 10-b and Fig. 10-c), *SQUINT* is neither.

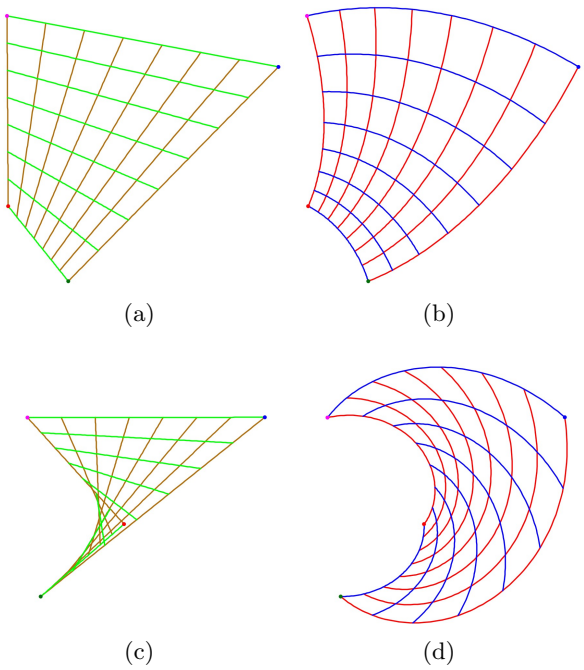


Figure 9: Bilinear map (a) and *SQUINT* map (b) for the same control corners. For a different configuration of control corners, the Bilinear map folds over itself (c), but the *SQUINT* map does not.

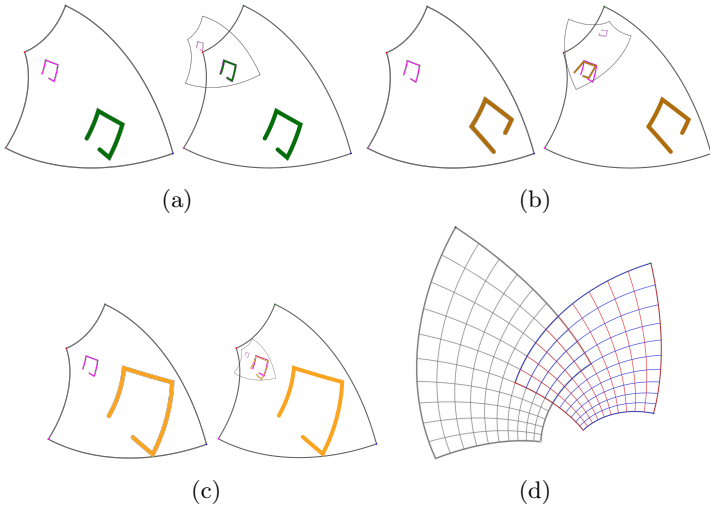


Figure 10: *SQUINT* map is transSimilar (a), but not rotSimilar (b), not dilSimilar (c). TranSimilarity makes it possible to overlap two similarity copies of a *SQUINT* map so that the tiles in the overlap region match perfectly (d).

3.5 Inverse and invertibility

To invert the *SQUINT* map, given a query point $\mathbf{q}$, we want to compute parameter pair (u, v) such that $\mathbf{q} = \mathbf{m}(u, v)$ (Fig. 11-a). We denote this operation by $(u, v) = \mathbf{m}^{-1}(q)$. Hence, we want to solve the following point-equation in u and v, $\mathbf{q} = \mathbf{f} + \lambda_u^u \lambda_v^v \overrightarrow{\mathbf{fa}}^{\circ}(u\alpha_u + v\alpha_v)$, or equivalently the vector equation, $\overrightarrow{\mathbf{fq}} = \lambda_u^u \lambda_v^v \overrightarrow{\mathbf{fa}}^{\circ}(u\alpha_u + v\alpha_v)$. It can be written $\overrightarrow{\mathbf{fq}} = \lambda_u^u \lambda_v^v (\cos(u\alpha_u + v\alpha_v)\overrightarrow{\mathbf{fa}} + \sin(u\alpha_u + v\alpha_v)\overrightarrow{\mathbf{fa}}^{)}$. This is a non-linear system of two simultaneous equations in two variables, u and v. We reduce it to a system of two simple linear equations as follows. We define $\lambda = |\mathbf{fq}| \div |\mathbf{fa}|$ and $\alpha = \overrightarrow{\mathbf{fa}} \wedge \overrightarrow{\mathbf{fq}}$. Then, we use the observation that, for a given (u, v) pair, the *SQUINT* map works as a similarity and performs a dilation with respect to $\mathbf{f}$ by $\lambda_u^u \lambda_v^v$ and a rotation around $\mathbf{f}$ by $u\alpha_u + v\alpha_v$. We obtain a decoupled system of two linear equations: $\alpha = u\alpha_u + v\alpha_v$, which matches the rotation angles, and $\lambda = \lambda_u^u \lambda_v^v$, which matches the dilation factors and can also be written as $\log \lambda = u \log \lambda_u + v \log \lambda_v$. The determinant of this system is $d = \alpha_v \log \lambda_u - \alpha_u \log \lambda_v$. There is no solution when $d = 0$. This may happen when the quad is a parallelogram (Fig. 11-b), in which case we can find the inverse easily using the Bilinear model, or when the *SQUINT* range $\mathbf{R}$ collapses to a segment of a log-spiral curve (Fig. 11-c) (or even to a point), in which case the inverse is not defined. Otherwise, we set $u = (\alpha_v \log \lambda - \alpha \log \lambda_v)/d$ and $v = (\alpha - u\alpha_u)/\alpha_v$.

3.6 Log-polar coordinates and extrapolation

The *SQUINT* map $\mathbf{m}(u, v)$ is the composition, $\mathcal{P} \cdot \mathcal{L}$ of a log-polar transform, $\mathcal{P}$, and a linear transform, $\mathcal{L}$. Specifically, $(\alpha, \rho) = \mathcal{L}(u, v)$, where $\mathcal{L}(u, v) = (u\alpha_u + v\alpha_v\,,\, u \ln \lambda_u + v \ln \lambda_v)$ and $(x, y) = \mathcal{P}(\alpha, \rho)$, where $\mathcal{P}(\alpha, \rho) = \mathbf{e}^{\rho} \overrightarrow{\mathbf{fa}}^{\circ} \alpha$. Hence, the inverse of the *SQUINT* map may equivalently be obtained using $(u, v) = \mathcal{L}^{-1} \cdot \mathcal{P}^{-1}(q)$.

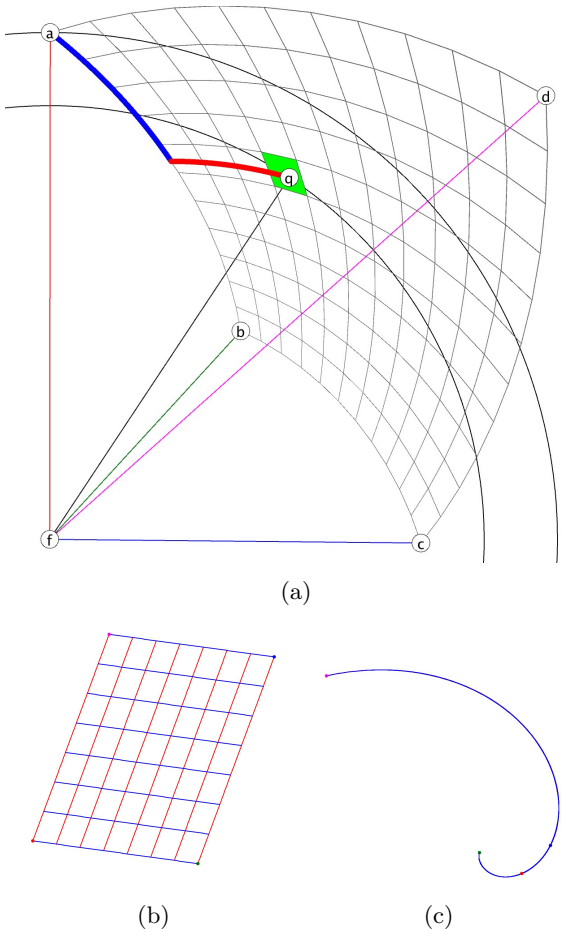
For sake of simplicity and elegance, we have, so far, restricted the *SQUINT* map to operate on parameter pairs in the unit-square domain, $\mathbf{D}$. However, the *SQUINT* map is defined outside of $\mathbf{D}$. Extrapolation is easy because the closed-form expressions of both the *SQUINT* map and its inverse work outside of the domain, with the caveat (discussed above) and ambiguity of angles α_u and α_v being defined modulo 2π.

For references, in Fig. 12, we draw the isocurves of $\mathbf{R}$ in darker lines over a lighter depiction of those in its extrapolation, $\mathbf{R}^+$. The example in Fig. 12-a suggest a wave and may be useful for aesthetic design or animation applications. In the example of Fig. 12-b, $\mathbf{R}^+$ closes on itself. The control corners have been adjusted to ensure that adjacent tile borders match along the junction. In Fig. 12-c, $\mathbf{R}^+$ is extended further and self-overlaps in a seamless manner. In Fig. 12-d, we use such an extrapolation to produce a steady two-pattern of disks.

3.7 Pseudo-conformal and conformal maps

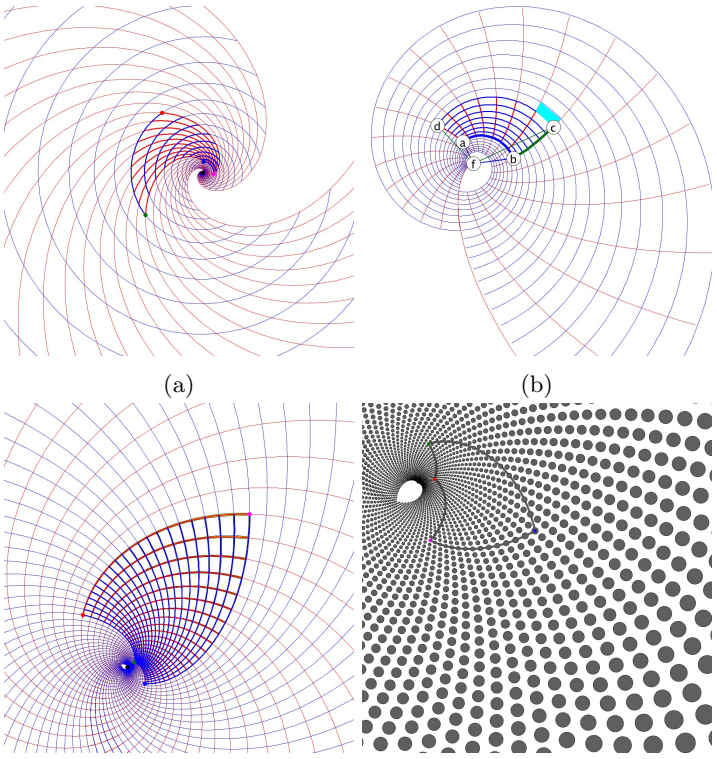
Because of tranSimilarity, the tiles of a *SQUINT* map are similar to each other. Because the iso-curves are tangent-continuous, the opposite corners of any given tiles have identical angle measures.

By foregoing one degree of freedom, we can constrain the arrangements of control corners, so as to ensure that the angles at all four corners of each tile are right angles. This happens when the following constraint is satisfied: $\tan^{-1}(\alpha_u / \ln \lambda_u) -$



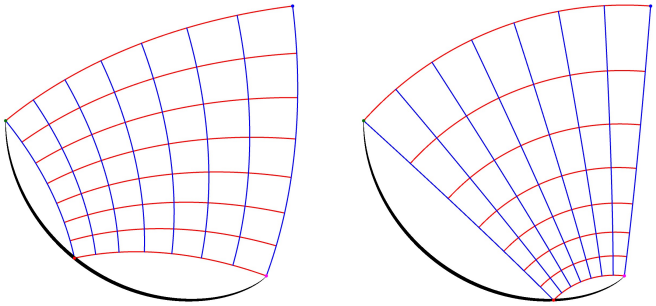
(a) (b) (c)

Figure 11: The parameters (u,v) of query point **q** are illustrated (a) by blue and red curves along the path from corner **a** to **q**. Two configurations for which the linear system does not produce an inverse are shown: The control quad is a parallelogram (b) and the four control points are on the same log-spiral curve (c).



(a) (b) (c) (d)

Figure 12: The tiling of **R** (thick curves) is drawn (a) over an extrapolation, $\mathbf{R}^+$, of the *SQUINT* map to the domain $\mathbf{D} = [0,1]^2$. The same extrapolation (b) is shown for a configuration where $\mathbf{R}^+$ almost closes on itself around **f**, except for a small gap. A tweak of that configuration (c) produces an $\mathbf{R}^+$ that self-overlaps. A careful adjustment of the control corners was used to ensure a seamless overlap. The same technique may be used (d) to design extended two-patterns of shapes. The boundary of **R** is drawn (dark line) for reference.



(a) (b)

Figure 13: We show two different configurations of the *SQUINT* map, for which the tiles have all four angles equal to $\pi/2$. For each configuration, we also show (thick black stroke) the curve along which we could slide **a** while maintaining this property.

$\tan^{-1}(\alpha_v / \ln \lambda_v) = \pi/2$. In such configurations, we say that the *SQUINT* map is ***pseudo-conformal***. In Fig. 13, we show (as a thick black stroke), for a configuration of the three control corners **b**, **c**, and **d**, the curve where control corner **a** should lie so as to make the *SQUINT* tiles all have right angles.

A pseudo-conformal map is conformal in symmetric configurations, such as the one shown in (Fig. 14-a). But it is not conformal in more general configurations. For example, in Fig. 14-c, we trace green and brown diagonals, which intersect at right angle in parameter-space, but not in the image space. Still, we suggest that the *SQUINT* map may provide a useful approximation or alternative for some conformal maps, as shown in Fig. 14-b, where we fit a *SQUINT* map (shown using blue and red isocurves) so that its control corners match those of a portion of a conformal map.

Harmonic and conformal maps are often used in graphics for parameterizing or flattening a curved surface, while minimizing distortion [2]. But, for some applications, the goal of minimizing local distortion may be less important than the goal of distributing the distortion uniformly. For those, the *SQUINT*

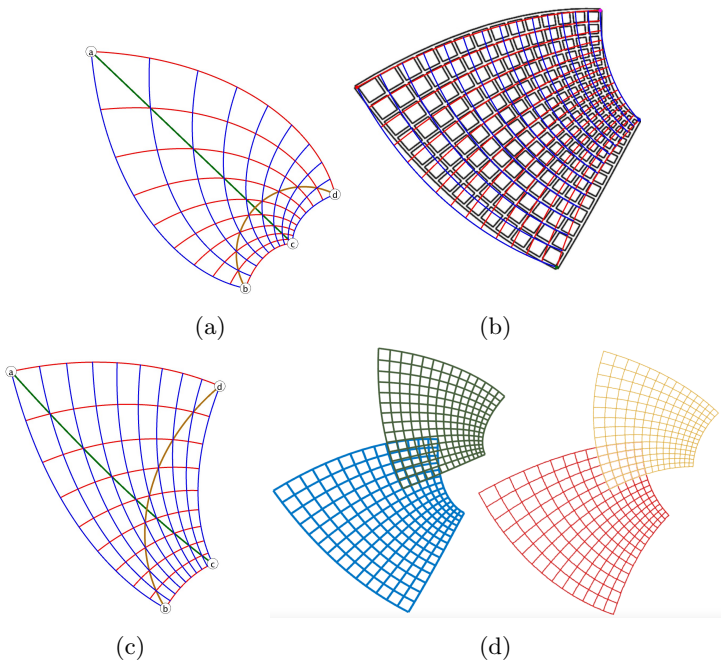


Figure 14: Conformal *SQUINT* map (a), pseudo-conformal *SQUINT* map (c), use of a *SQUINT* map to approximate a conformal map (from `https://en.wikipedia.org/wiki/Conformal_map`) (b), and an example (d) showing that a conformal map is not self-similar (d-left) while its approximating *SQUINT* map (d-right) is.

map may prove more useful, as it is self-similar (Fig. 14-d) and distributes the distortion evenly over the entire domain (Fig. 9-a and b).

It is informative to compare the mathematical formulations and properties of the *SQUINT* map to those of the ***Moebius transformations***, which is defined over the complex plane by $f(z) = (az + b)/(cz + d)$, where z, a, b, c, and d are complex numbers with $ad \neq bc$. Note that $z + b$ corresponds to a translation and $az + b$ to a similarity. If we use $z = x + iy$ to represent the point with coordinates (x, y), the numerator and denominator of the Moebius transform define similarity transformations that operate on that point. In contrast, although *SQUINT* also uses two similarities, it uses x and y as powers to select how much of each transformation to use. The Moebius transformation has two fixed points. In contrast, *SQUINT* map has a single fixed point, **f** . The Moebius transform may be controlled by three points. In contrast, *SQUINT* offers four control points. As *SQUINT* does, the Moebius transform also has a simple inverse: $f^{-1}(z) = (dz - b)/(a - cz)$. The Moebius transformation preserves angles and circles. *SQUINT* does not, but may be used to produce perfect circles. In spite of these profound differences, portions of the grid of iso-curves of the Moebius transformation may often be closely approximated by a *SQUINT*. But the matching is not perfect, hence, *SQUINT* might be considered as an approximation of (or an alternative to) the Moebius transform, but not as its generalization.

3.8 Scalar and vector fields

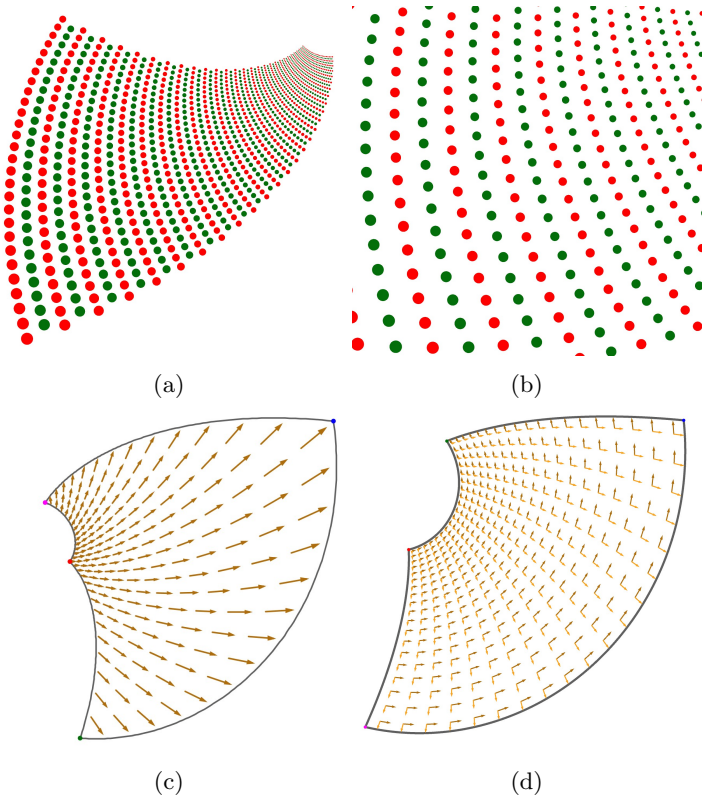


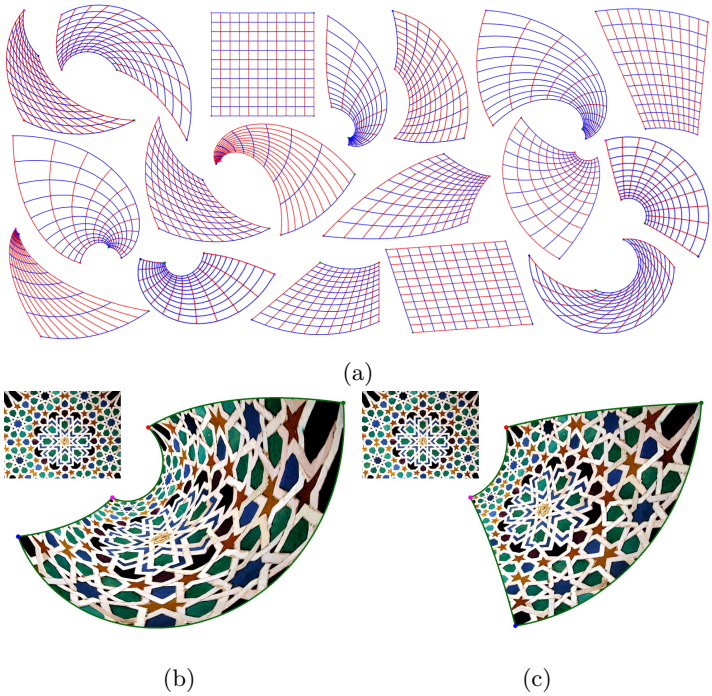
Figure 15: Two *SQUINT* patterns (red and green) of a disk (a) and a detail (b) illustrate the use of the *SQUINT* map to control simple scalar fields. *SQUINT* may be used to control a vector field (c) and (d).

We can use *SQUINT* to control a simple scalar field. We show such a field using green and red disks of different sizes in Fig. 15-a and -b. Such simple fields may be useful to control the gentle gradation of material density in a structure. When displaying the disk for sample (u, v), the chosen radius of the template disk is scaled by $\lambda_u^u \lambda_v^v$. We can also use *SQUINT* to control a gently varying, curved vector field (Fig. 15-c and -d). Such a field may be used to guide flocking motions or smoke animations or to control the orientation of short fibers in a material. Note that, in addition to the overall bending of the tile structure, the designer may rotate simultaneously all the vectors around their origin.

3.9 Versatility and texture mapping

Both the bilinear and *SQUINT* interpolations have the same number of degrees of freedom and may be controlled in a similar manner by the four corner points. Given that it has only as many degrees of freedom as a quadrilateral, *SQUINT* can produce an impressive (although of course limited) range of shapes (Fig. 16-a). We believe that *SQUINT* might be useful as a primitive tool for creating simple, yet useful texture mapping effects (Fig. 16-b and -c) and animations of images or fields that

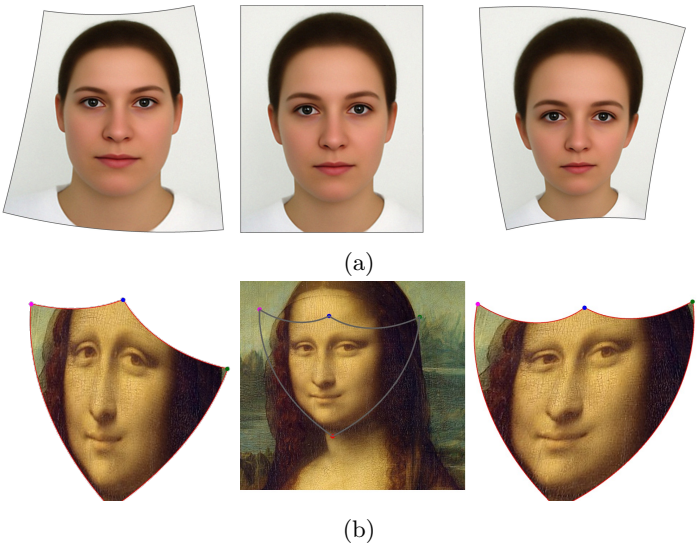
are aesthetically pleasing, because the deformation is extremely regular, yet not boringly linear.



(a)

(b) (c)

Figure 16: Examples of possible deformations produced by *SQUINT* (a) and its use to deform a picture of an Islamic texture in different ways (b and c).



(a)

(b)

Figure 17: Two different *SQUINT* maps (a) shown (left and right) of the image of a face (center), with permission from [1]. A composition of a *SQUINT* map and of the inverse of another used to select and deform a portion of Mona's face (b).

SQUINT may be used to map (as texture) a square or rectangular image onto a distorted planar shape. An example is offered in (Fig. 17-a), where we show two *SQUINT* deformations of the same face. We show (Fig. 2-d) an attempt to exploit steadiness for an artistic effect. Because of the efficient and precise inversion, *SQUINT* may be useful for the precise rendering of deformed textures by computing the exact texture coordinates for the center of each pixel in **R** during its rasterization. We have also explored the use of a cascade of two *SQUINT* maps. The **R** shape of the first one is edited by the artist on the image of the texture and used as a selection tool for a curved-quad region around an area of interest. The **R** shape of the second one is edited independently to specify where that cut-out region should be placed in the final composition and how it should be deformed. Hence, the mapping from texture to final image is the composition of a *SQUINT* with the inverse of another *SQUINT*. We show its use to change the mood and gaze of Mona Lisa (Fig. 17-c).

3.10 Non-steady symmetric bi-spiral map

For some applications, one may trade steadiness for additional freedom. Fig. 18 shows three different configuration of a symmetric bi-spiral map extension of the *SQUINT* map. In this extension, in addition to the four control corners, **a**, **b**, **c**, and **d**, we let the designer also control four other control points, $\mathbf{a}'$, $\mathbf{b}'$, $\mathbf{c}'$, and $\mathbf{d}'$. We show the vectors $\overrightarrow{\mathbf{aa}'}$, $\overrightarrow{\mathbf{bb}'}$, $\overrightarrow{\mathbf{cc}'}$, $\overrightarrow{\mathbf{dd}'}$ as blue arrows. We build upon the bilinear interpolation $\mathbf{i}(\mathbf{i}(\mathbf{a},u,\mathbf{b}),v,\mathbf{i}(\mathbf{d},u,\mathbf{c}))$, but replace each LERP by the steady morph between the corresponding arrows. For example, we replace $\mathbf{i}(\mathbf{a},u,\mathbf{b}) = (1-u)\mathbf{a}+u\mathbf{b}$ by the image of oriented edge $\mathbf{aa}'$ by $\mathcal{U}^t$ where $\mathcal{U} = \mathrm{Sim}(\mathbf{a},\mathbf{a}',\mathbf{b},\mathbf{b}')$. In fact, to make the solution symmetric, we perform the evaluation twice (swapping the roles of the u and v directions) and then blend these slightly different solutions by using the mid-course of a steady morph between each pair of two corresponding edges. Based on extensive interactive manipulation of these eight control points and on their simultaneous animation along different spirals, each at a different frequency, we conjecture that this extension is stable and produces smoothly varying and visually pleasing tilings and tiling animations. However, these tilings are not steady, and hence do not have the properties of *SQUINT* maps. More importantly, two-patterns and lattices defined in terms of these ***symmetric bi-spiral map*** are more expensive to query than the corresponding patterns and lattices derived from the *SQUINT* map defined by control corners **a**, **b**, **c**, and **d** alone.

4 *SQUINT* PATTERN

4.1 Definitions

Based on the Fields of Similarities introduced above, we define two-index arrangements of similar copies $\mathbf{X}_{\mathrm{i,j}}$ of a template shape, $\mathbf{X}_{0,0}$. We call them ***two-patterns***. We denote such a two-pattern by $\{\mathbf{X}_{\mathrm{i,j}}\}$. For simplicity, we assume that the indices, i and j, are each integers in [0,n]. Hence, $\{\mathbf{X}_{\mathrm{i,j}}\}$ comprises $(\mathrm{n}+1)^2$ shapes. The extension of these results to two-patterns with different repetition counts for each index, i.e., where (i,j)

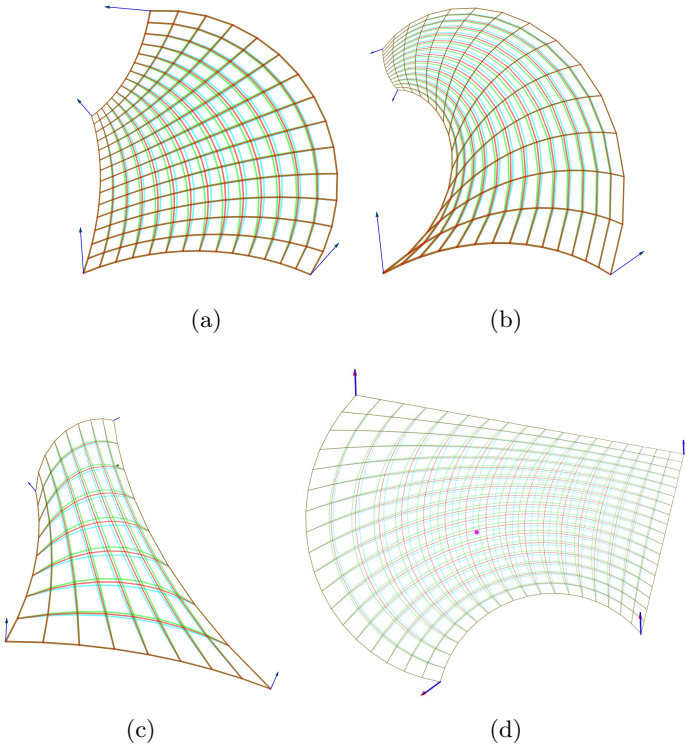


Figure 18: Examples of a symmetric bi-spiral map controlled by four corner frames, each defined by a point-vector pair or equivalently by an oriented edge. The two primary (u-first, and v-first) constructions are shown in green and cyan. Their average is shown in brown.

is in [0,m]×[0,n], is straightforward.

$\{\mathbf{X}_{i,j}\}$ is a ***regular pattern*** when there exist two vectors, $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$, such that $\mathbf{X}_{i,j} = \mathbf{X}_{0,0}+i\vec{\mathbf{u}}+j\vec{\mathbf{v}}$. Note that the instances of a regular pattern are not only similar, but identical, translated copies of $\mathbf{X}_{0,0}$ and that they form a parallelogram arrangement.

$\{\mathbf{X}_{i,j}\}$ is a ***PSCS pattern*** when there exists a template, $\mathbf{X}_{0,0}$, a repetition count, n, and a PSCS (Product of Steady Curves of Similarities), $\mathcal{S}_{u,v}$, such that $\mathbf{X}_{i,j} = \mathcal{S}_{i/n,j/n} \cdot \mathbf{X}_{0,0}$.

$\{\mathbf{X}_{i,j}\}$ is a ***steady pattern*** when there exists a template, $\mathbf{X}_{0,0}$, a repetition count, n, and an SFS (Steady Field of Similarities), $\mathcal{S}_{u,v}$, such that $\mathbf{X}_{i,j} = \mathcal{S}_{i/n,j/n} \cdot \mathbf{X}_{0,0}$.

$\{\mathbf{X}_{i,j}\}$ is a ***SQUINT pattern*** when it is a steady pattern for which $\mathcal{S}$ is a *SQUINT* field. Hence it is defined by four different control corners **a**, **b**, **c**, and **d**.

4.2 Self-similarity

Consider four neighboring elements in a *SQUINT* pattern: $\mathbf{X}_{i,j}$, $\mathbf{X}_{i+1,j}$, $\mathbf{X}_{i,j+1}$, and $\mathbf{X}_{i+1,j+1}$ By definition, we have $\mathbf{X}_{i,j} = \mathcal{U}^{i/n} \cdot \mathcal{V}^{j/n} \cdot \mathbf{X}_{0,0}$. The following derivations show that $\mathbf{X}_{i+1,j}$, $\mathbf{X}_{i,j+1}$, and $\mathbf{X}_{i+1,j+1}$ are each related to $\mathbf{X}_{i,j}$ by a constant similarity, i.e., one that is independent of the values of indices i and j: $\mathbf{X}_{i+1,j} = \mathcal{U}^{(i+1)/n} \cdot \mathcal{V}^{j/n} \cdot \mathbf{X}_{0,0} = \mathcal{U}^{1/n} \cdot (\mathcal{U}^{i/n} \cdot \mathcal{V}^{j/n} \cdot \mathbf{X}_{0,0}) = \mathcal{U}^{1/n} \cdot \mathbf{X}_{i,j}$, $\mathbf{X}_{i,j+1} = \mathcal{U}^{i/n} \cdot \mathcal{V}^{(j+1)/n} \cdot \mathbf{X}_{0,0} = \mathcal{V}^{1/n} \cdot (\mathcal{U}^{i/n} \cdot \mathcal{V}^{j/n} \cdot \mathbf{X}_{0,0}) = \mathcal{V}^{1/n} \cdot \mathbf{X}_{i,j}$, and $\mathbf{X}_{i+1,j+1} = \mathcal{U}^{(i+1)/n} \cdot \mathcal{V}^{(j+1)/n} \cdot \mathbf{X}_{0,0} = \mathcal{V}^{1/n} \cdot \mathcal{U}^{1/n} \cdot (\mathcal{U}^{i/n} \cdot \mathcal{V}^{j/n} \cdot \mathbf{X}_{0,0}) = \mathcal{V}^{1/n} \cdot \mathcal{U}^{1/n} \cdot \mathbf{X}_{i,j}$. As a consequence, a *SQUINT* pattern is self-similar. By this, we mean that a non-trivial similarity exists that aligns a copy of the *SQUINT* pattern such that the overlapping elements match perfectly (Fig. 19).

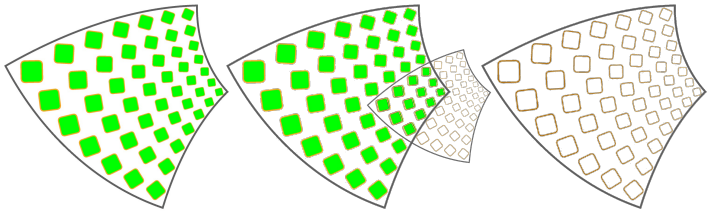


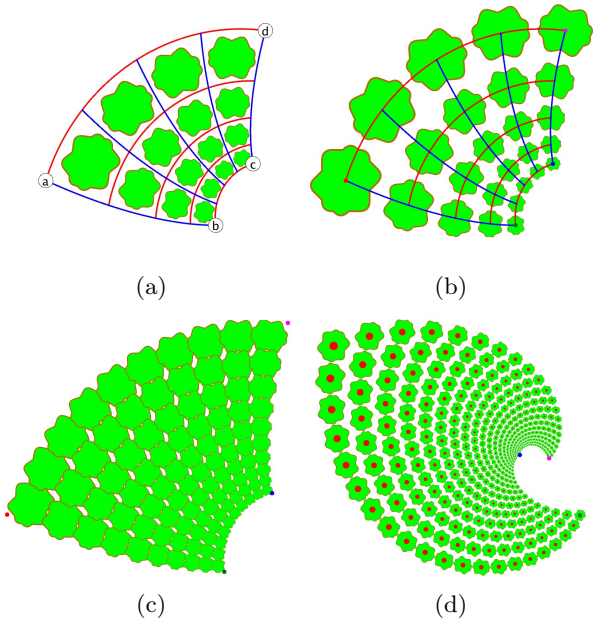
Figure 19: The *SQUINT* pattern $\{\mathbf{P}\}$ has 7×7 green elements (left). A copy $\{\mathbf{P}'\}$ of $\{\mathbf{P}\}$ is shown translated (right) with unfilled elements. We then rotate $\{\mathbf{P}'\}$ counterclockwise a bit, scale it down, and translate it so that its left 3×4 portion matches perfectly the right 3×4 portion of $\{\mathbf{P}\}$ (center).

4.3 *SQUINT* patterns of shapes and of tiles

In an n×n two-pattern of tiles of a *SQUINT* map, the base tile $\mathbf{T}_{0,0}$ is the image by **m** of the base square, $\mathbf{D} = [0, 1/n]^2$ (a corner cell in a rectilinear (n+1)×(n+1) lines grid-partition of D). With $\mathcal{M} = \mathcal{U}^{1/n}$ and $\mathcal{N} = \mathcal{V}^{1/n}$, the tiles may be formulated as $\mathbf{T}_{i,j} = \mathcal{N}^j \cdot \mathcal{M}^i \cdot \mathbf{T}_{0,0}$. $\mathcal{M}$ and $\mathcal{N}$ commute because $\mathcal{U}$ and $\mathcal{V}$ do. Index pair (i, j) lies in $[0, 1]^2$. Hence, $\mathbf{T}_{i+1,j} = \mathcal{N}^j \cdot \mathcal{M}^{i+1} \cdot \mathbf{T}_{0,0} = \mathcal{M} \cdot (\mathcal{N}^j \cdot \mathcal{M}^i \cdot \mathbf{T}_{0,0}) = \mathcal{M} \cdot \mathbf{T}_{i,j}$. Similarly, we can show that $\mathbf{T}_{i,j+1} = \mathcal{N} \cdot \mathbf{T}_{i,j}$ and that $\mathbf{T}_{i+1,j+1} = \mathcal{N} \cdot \mathcal{M} \cdot \mathbf{T}_{i,j}$, In conclusion, the tiles of *SQUINT* map form a *SQUINT* pattern. Even though the tiles are similar to each other, they are not, in general, similar to **R**.

In Fig. 20-a, we show a 4×4 *SQUINT* patterns of tiles with a 4×4 *SQUINT* pattern of a green template shape $\mathbf{X}_{0,0}$ that is aligned so that each green element fits inside a tile. In Fig. 20-b, we show the same *SQUINT* pattern of tiles, but use it to define a 5×5 *SQUINT* pattern of that same green shape, where the elements are roughly centered at the grid-crossings. The latter choice was used in [4, 7].

Note that, not only are all green elements $\mathbf{X}_{i,j}$ similar to the template $\mathbf{X}_{0,0}$ and hence to their neighbors, but the overlaps and gaps between them are also similar to each other and form steady (actually *SQUINT*) patterns. This regularity is more blatant in Fig. 20-c and -d. The image $\mathbf{X}(u, v) = \mathcal{S}_{u,v} \cdot \mathbf{X}$ of template shape, $\mathbf{X}_{0,0}$, transformed by $\mathcal{S}_{u,v}$ may be displayed by executing the following instructions, where point **o** is the global origin: translate($\overrightarrow{\mathbf{of}}$); rotate($u\alpha_u + v\alpha_v$); scale($\lambda_u^u \lambda_v^v$); translate($\overrightarrow{\mathbf{fo}}$); display($\mathbf{X}$); With the choice of displaying an n×n grid of elements, one per tile, u = i/n and v = j/n. With the choice of displaying an (n+1)×(n+1) grid of elements, one per crossing, u = i/(n + 1) and v = j/(n + 1).



(a) (b) (c) (d)

Figure 20: Control corners **a**, **d** , **b** , and **c**, are used to control the overall shape and geometric gradation of these *SQUINT* patterns. We show a 4×4 *SQUINT* pattern of green shapes that each fit inside a tile (a), 5×5 *SQUINT* pattern where the elements are centered on grid-crossings (b and d). Observe that the gaps (d) and overlaps (c) between adjacent tiles form *SQUINT* patterns.

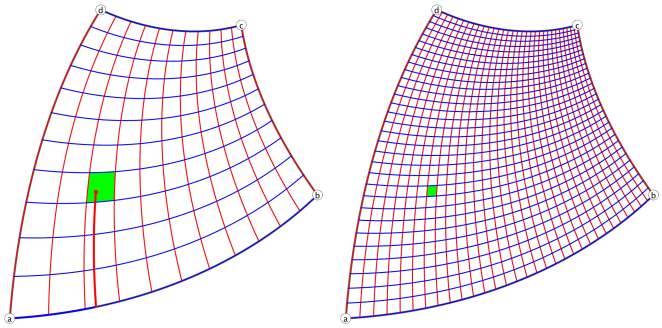
4.4 Area computation of a *SQUINT* pattern

For simplicity, assume first that each element of an n×n steady pattern $\{\mathbf{X}_{i,j}\}$ fits in a separate tile of the corresponding *SQUINT* pattern, as illustrated in Fig. 20-a. Let x denote the area of the base shape, $\mathbf{X}_{0,0}$, t denote the area of the base tile, $\mathbf{T}_{0,0}$, and r denote the total area of $\mathbf{R}$. The total area of $\{\mathbf{X}_{i,j}\}$ is $a = xr/t$. Hence, it may be computed in constant time, independently of the value of n. This surprisingly simple closed-form, constant-cost expression for the area of a *SQUINT* pattern can be extended to configurations illustrated in Fig. 20-b by suitably enlarging the domain $\mathbf{D}$ and splitting it into (n+1)×(n+1) tiles such that each $\mathbf{X}_{i,j}$ is fully contained in tile $\mathbf{T}_{0,0}$. It may be further extended to configurations where $\mathbf{X}_{i,j}$ is not fully contained in tile $\mathbf{T}_{0,0}$, provided that the shapes of $\{\mathbf{X}_{i,j}\}$ are pairwise disjoint. It may even be extended to configurations illustrated in Fig. 20-c, but such an extension is more complex, and requires special processing of corner tiles and of the four one-patterns of tiles along the border of $\mathbf{X}_{i,j}$.

4.5 Tile containing a query point

Given a query point $\mathbf{q}$, we can compute, in constant time (i.e., independent of n), the indices (i,j) of the tile, $\mathbf{T}_{i,j}$, that contains it. This provides a considerable saving over an $O(n^2)$ brute-force approach that would classify $\mathbf{q}$ against each tile. Furthermore, note that the approach proposed here does not work for the more general (non-steady) PSCS patterns. To compute the indices (i,j) of the desired $\mathbf{T}_{i,j}$, we first use the inversion, $\mathbf{m}^{-1}$, discussed in Section 3 to compute the parameter-pair, (u, v), such that $\mathbf{q} = \mathbf{m}(u, v)$. Then, we simply use the floor operator to compute the green tile indices: $i = \lfloor un \rfloor$ and $j = \lfloor vn \rfloor$.

Having a closed-form expression allows us to perform this calculation instantaneously and with high numeric accuracy. Fig. 21-a shows (thicker blue and red curves) the (u, v) parameters of $\mathbf{q}$ and fills, in green, the tile $\mathbf{T}_{i,j}$ that contains $\mathbf{q}$. The same point $\mathbf{q}$ identifies a different tile for the same *SQUINT*, but with a higher n (Fig. 21-b). This query may be useful to accelerate the Point Membership Classification against *SQUINT* patterns, because it identifies the indices of the element or elements associated with that cell or with neighboring ones.



(a) (b)

Figure 21: In green, the tile that contains the same query point $\mathbf{q}$ for different grid resolutions (b) and (b).

5 *SQUINT* LATTICE

In this section, we discuss the potential benefits of using *SQUINT* Patterns and *SQUINT* Maps for designing and querying parameterized procedural models of highly complex, planar, truss-like, lattice structures.

5.1 Disk and bar patterns

A *SQUINT* lattice is the union of d *SQUINT* ***patterns of disks*** (nodes) and of b *SQUINT* ***patterns of bars*** (quads), each bar connecting two disks with tangential continuity (so that the union of a bar with the two disks that it connects is the convex hull of these disks). We are given four different control corners (**a**, **b**, **c**, **d**) and the repetition count n. We first compute the global similarities: $\mathcal{U} =< \mathbf{f}, \lambda_u, \alpha_u >= \mathrm{Sim}(\mathbf{a}, \mathbf{d}, \mathbf{b}, \mathbf{c})$ and $\mathcal{V} =< \mathbf{f}, \lambda_v, \alpha_v >= \mathrm{Sim}(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{c})$. Then, we compute the incremental similarities: $\mathcal{M} = \mathcal{U}^{1/n} =< \mathbf{f}, \lambda_u^{1/n}, \alpha_u/n >$ and $\mathcal{N} = \mathcal{V}^{1/n} =< \mathbf{f}, \lambda_v^{1/n}, \alpha_v/n >$.

Then we are given an array, $\mathbf{D}[]$, of d ***disk templates***, where each template $\mathbf{D}[d]$ is defined by a center $\mathbf{o}[d]$ and radius $r[d]$. Typically, each disk template $\mathbf{D}[d]$ is placed in base tile $\mathbf{T}_{0,0}$ or near control corner **a**. The *SQUINT* pattern of disks associated with index d is $\{\mathbf{D}[i,j,d]\}$. The $[i,j]^{th}$ element of this two-pattern

is disk $\mathbf{D}$[i,j,d]=$\mathcal{N}^{\mathrm{j}} \cdot \mathcal{M}^{\mathrm{i}} \cdot \mathbf{D}$[d], which has center $\mathcal{N}^{\mathrm{j}} \cdot \mathcal{M}^{\mathrm{i}} \cdot \mathbf{o}$[d] and radius $\lambda_u^{\mathrm{i}} \lambda_v^{\mathrm{j}} r$[d]. So, we have a total of dn^2 disks.

Finally, we are given an array, $\mathbf{B}$[b], of b bar templates, each defined by the indices, s[b] and e[b], and by the index offsets, p[b] and q[b], that identify the start and end disks that the bar connects. The *SQUINT* pattern of bars associated with index b is {$\mathbf{B}$[i,j,b]}. The [i,j]$^{\mathrm{th}}$ element of this pattern is bar $\mathbf{B}$[i,j,b]=$\mathcal{N}^{\mathrm{j}} \cdot \mathcal{M}^{\mathrm{i}} \cdot \mathbf{B}$[b], which connects disks $\mathbf{D}$[i,j,s[b]] and $\mathbf{D}$[i+p[b],j+q[b],e[b]], but only for pairs [i,j] for which both disks exist.

5.2 The two-patterns of hubs of a clean lattice

We split each bar into two ***stumps***. For this, we use a splitting cut-edge that is orthogonal to the axis of the bar and equidistant from the disks that the bar connects. With each disk, $\mathbf{D}$[i,j,d], we associate ***hub***, $\mathbf{H}$[i,j,d], which is the union of $\mathbf{D}$[i,j,d] with the stumps that connect to it. For each d, {$\mathbf{D}$[i,j,d]} is a *SQUINT* pattern.

Let us, at first, ignore exceptions (which we discussed later) at border hubs. Because the incremental transformations in a *SQUINT* pattern are constant, for each b, {$\mathbf{B}$[i,j,b]} is also a *SQUINT* pattern. Hence, the stumps form *SQUINT* patterns. Consequently, for each d, {$\mathbf{H}$[i,j,d]} is a *SQUINT* pattern. The lattice is the union of *SQUINT* patterns of hubs.

The lattice is ***clean*** when the hubs are ***quasi-disjoint*** (i.e., when their interiors do not overlap, even though hubs with stumps of the same bar touch along the split-edge that divides that bar).

In Fig. 22-a, we show 4 *SQUINT* patterns of disks, each filled in a different color. For clarity, we ensured here that each ***group*** of 4 disks with the same index pair [i,j] fits in the corresponding tile, $\mathbf{T}_{\mathrm{i,j}}$, but this may not always be the case. In Fig. 22-b, we show 6 *SQUINT* patterns of bars. Each tile, $\mathbf{T}_{\mathrm{i,j}}$, contains four small bars, which form a diamond. These are the ***intra-group bars***: they join disks of the same group. Two patterns of longer bars leave disks of most groups to connect them to disks in neighboring groups, one in the i-direction and one in the j-directions. These are the ***inter-group bars***. Note that groups in the top row and in the right column are missing some of these inter-group bars, because they do not have neighboring groups in the corresponding directions. We discuss these and other exceptions in Subsection 5.5. The stumps are colored with the color of the disk that they connect to. This coloring helps the viewer distinguish the different hubs. (Notice the red, green, orange, and brown *SQUINT* patterns of hubs.) Keeping the same control corners and disk and bar definition, but increasing the repetition count n, produces a finer lattice (Fig. 22-c). Note that, when n is changed, we automatically adjust the disk radii an positions so that their size relative to the base tile remains the same. We show (Fig. 22-d) a more distorted version of the 4×4 lattice (Fig. 22-b).

In (Fig. 23-a) we show a clean lattice, in which the brown disk has been moved so that it no longer fits in the same tiles as the other three disk of its group. This configuration illustrates the fact that a terminology based on groups and hubs is more general and elegant than one based on ***cells***, which are often defined as the intersection of the lattice with a tile. Observe for example that an interior cell of the *SQUINT* lattice in Fig. 23-a is disconnected (it has three maximally connected components, two of which are brown). Using the same disks and bars, but moving the control corners may create an ***unclean lattice*** (Fig. 23-b). But because of steadiness, the offending interference between different hubs occurs systematically between all hubs of non-border groups. Hence, these ***interferences may be detected in constant time***. We also show that a clean lattice (Fig. 23-c) may become unclean (Fig. 23-d), even if we keep its control corners constant, but decrease the repetition count. Again, the good news is that, for a *SQUINT* lattice, such interferences may be tested in constant time. In the remainder of the paper, we will ***assume that the lattice is a clean SQUINT lattice***.

In Fig. 23 and 22, we use the n×n option, with each element of a given disk pattern corresponding to a different tile. Fur-

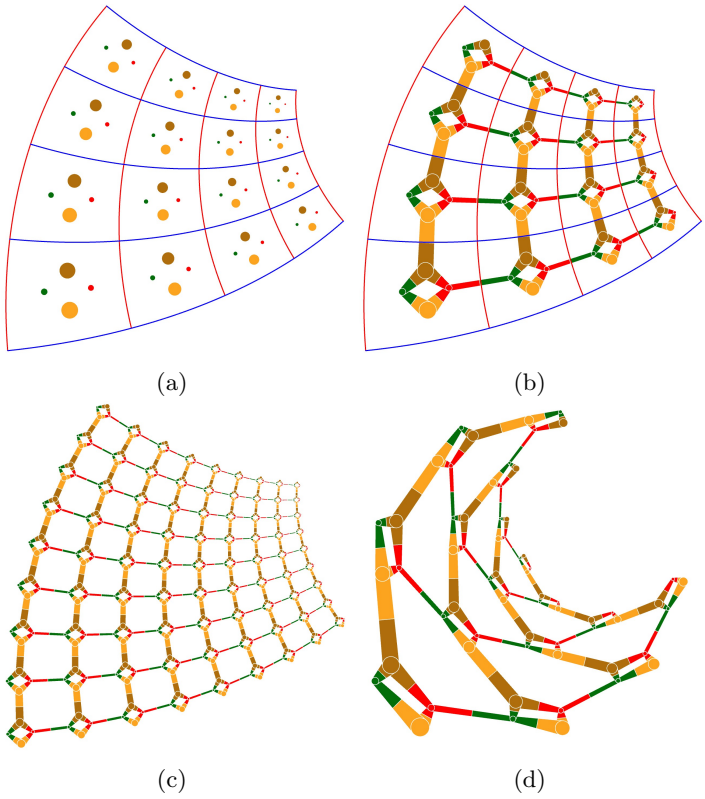


Figure 22: Four *SQUINT* patterns of disks (red, green, orange, and brown) form a 4×4 *SQUINT* pattern of disk groups (a). Six *SQUINT* patterns of bars (four of which are intra-group) are show (b) each split into two stumps that are painted with the color of the disk that they connect to. The hubs are quasi-disjoint, hence this lattice is clean. The hub of each color form a *SQUINT* patterns. A refined lattice (c) is obtained by increasing n. It retains all the above properties. The lattice (d) was obtained from the 4×4 lattice (b) by moving the control corners.

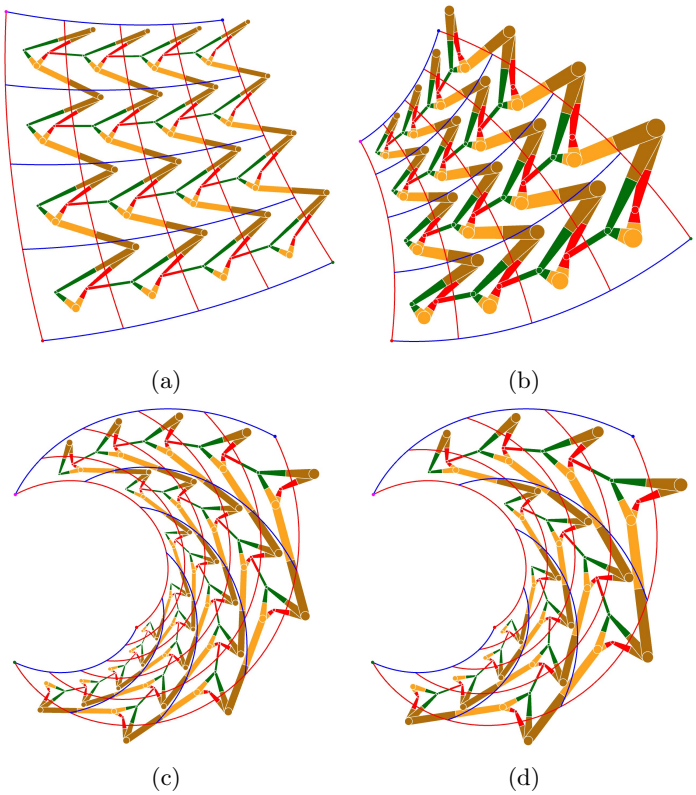


Figure 23: A clean *SQUINT* lattice (a) and an unclean version obtained by moving the control corners. A clean *SQUINT* lattice (c) and an unclean version obtained by reducing the repetition count.

thermore, we register the preimage of the center of each disk and its radius in parameter space relative to the base tile. This choice allows us to edit the entire lattice using only the four control corners and to trivially change the repetition counts. Furthermore, it also ensures that pairs of disk centers that were originally placed along an isocurve (for example the centers of the orange and brown disks of the same or adjacent groups in 22) remain on the same isocurve when the SQUINT map is edited.

In contrast, in [4, 7] the (n+1)×(n+1) option was chosen and the transformations $\mathcal{U}$ and $\mathcal{V}$ were specified directly, either by programming instructions that define them or by using the graphics interface to position, orient, and scale the [0,0], [n,0], and [0,n] groups of disks. Such a paradigm may be preferred when the orientation and shape of disk group [0,0] needs to be controlled explicitly, independently of $\mathcal{U}$ and $\mathcal{V}$.

5.3 Direct calculation of the area of a lattice

An effective strategy for computing the total area of a *SQUINT* lattice is to sum the areas t_{d} for each hub pattern d. Remember that we assume that the lattice is clean, so the hubs are pairwise disjoint. Furthermore, given the area a_{d} of a template hub $\mathbf{H}$[0,0,d] in hub pattern d, the area of hub $\mathbf{H}$[i,j,d] is $\lambda_v^{2\mathrm{j}/\mathrm{n}}\lambda_u^{2\mathrm{i}/\mathrm{n}}a_{\mathrm{d}}$ and the total area t_{d} is the product of two geometric progression sums and may be computed directly, without iterations, using the closed-form expression:

$$t_{\mathrm{d}} = \sum_{\mathrm{j}=0}^{\mathrm{n}-1} \lambda_v^{2\mathrm{j}/\mathrm{n}} \left(\sum_{\mathrm{i}=0}^{\mathrm{n}-1} \lambda_u^{2\mathrm{i}/\mathrm{n}} a_{\mathrm{d}} \right) = a_{\mathrm{d}} \left(\frac{\lambda_v^2 - 1}{\lambda_v^{2/\mathrm{n}} - 1} \right) \left(\frac{\lambda_u^2 - 1}{\lambda_u^{2/\mathrm{n}} - 1} \right)$$

A similar closed-form computation has been proposed in [4] for 3D lattices made of three-patterns of balls (instead of disks) and of cone-bars (instead of isosceles trapeze bars). For such ***slab lattices*** in which each ball pattern is a Product of Steady Curves of three-dimensional Similarities (defined by field $\mathcal{S}_{u,v} = \mathcal{V}^v \cdot \mathcal{U}^u$, for which $\mathcal{U}$ and $\mathcal{V}$ do not commute), that prior art computes the total volume of a slab lattice with a computational cost of O(n). As an extension to brick lattices, in which each ball pattern is a Product of Steady Curves of three-dimensional Similarities (defined by field $\mathcal{S}_{u,v,w} = \mathcal{W}^w \cdot \mathcal{V}^v \cdot \mathcal{U}^u$, for which none of the pairs commute), this prior art computes the total volume of the lattice with a computational cost of $\mathrm{O}(\mathrm{n}^2)$. Although our result presented above is limited to planar slab lattices that are constrained to be steady, the benefits of the reduction of computational cost that it affords may be of importance for the optimization of highly complex 2D lattices, in which the total area must be maintained constant, while the other lattice descriptors (control corners, repetition count, disks and bars) are varied.

SQUINT affords a different, constant cost approach for computing the total area of a *SQUINT* lattice, $\mathbf{L}$. It follows directly from the result presented in Subsection 4.4. First, assume that the intersections $\mathbf{L} \cap \mathbf{T}_{\mathrm{i,j}}$ form a steady pattern (no exceptions). Let x denote the area of $\mathbf{L} \cap \mathbf{T}_{0,0}$, t the area of $\mathbf{T}_{0,0}$ and r the total area of $\mathbf{R}$. The total area of $\mathbf{L}$ is $a = xr/t$. Hence, it may be computed in constant time, independently of the value of n. In practice, the above assumption is often valid only for the (n-2)×(n-2) block of interior (not border) tiles, see Fig. 22-c for an example. The four corner tiles and the four one-patterns of border tiles are exceptions that must be processed separately. But still, in such relatively simple configurations, this approach yields a constant cost solution that is reasonably simple to implement, especially if it is acceptable to use approximations of x, r, and t that are computed by counting pixels on the GPU for appropriately zoomed images of $\mathbf{T}_{0,0}$. r may be computed by standard integration over the boundary of $\mathbf{R}$ in constant time. Computing the area of lattices with more complex exception patterns, for example those discussed in Subsection 5.5, may conceivably, involve a non-constant number of special cases, and hence may have non-constant cost.

5.4 Point Membership Classification

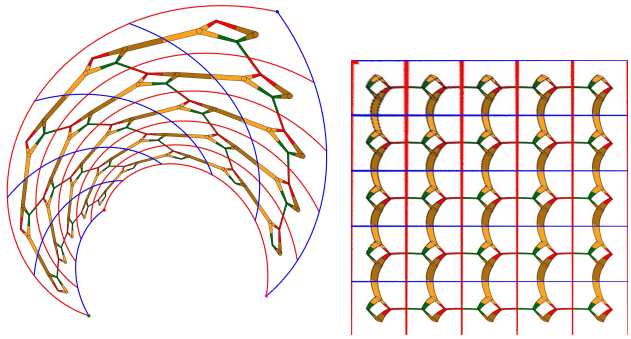
Point-Membership Classification (PMC) is a fundamental query for processing models of lattices and more generally assemblies of solids [5]. Given a lattice, $\mathbf{L}$, and query point, $\mathbf{q}$, we want to establish whether or not $\mathbf{q}$ lies inside any of the elements of $\mathbf{L}$. For a slab lattice, brute-force approach compares $\mathbf{q}$ to each

element, and hence has a computational time-cost of $O(n^2)$. Although various spatial-occupancy data structures may be precomputed to accelerate a sequence of these queries, their computation is expensive in time and space and useful only when it is amortized over a long sequence of queries. The acceleration of PMC for lattices has been addressed in prior art. In [15] a constant cost solution is proposed, but it is restricted to rectilinear patterns or to their warps. In [7], a O(n) time-cost solution is proposed for three-dimensional slab lattices that are a Product of Steady Curves of three-dimensional similarities. That solution exploits the fact that, at least in one parametric direction, the slab is a Steady Curves of three-dimensional similarities. It uses a canonical decomposition of that similarity into a commutative product of base 3D similarities. It computes the inversion of $\mathbf{q}$ for each one of these base similarities. Finally, it uses the resulting parameters to identify the range of [i,j] index-pairs for which a ball or bar might contain (i.e., is not guaranteed to not contain) $\mathbf{q}$. The solution proposed below builds on that approach, but, because it assumes that the lattice is a *SQUINT* lattice, it exploits the commutativity of $\mathcal{U}$ and $\mathcal{V}$ and uses the inversion of $\mathbf{m}$ to reduce the computational cost of PMC to a constant.

Several variants of this idea may be useful. For example, one may classify $\mathbf{q}$ against each *SQUINT* pattern of balls and against each *SQUINT* pattern of bars. Or it may be advantageous to classify $\mathbf{q}$ against each *SQUINT* pattern of hubs. But for conciseness, we describe here a simple variant. We will first ignore the connectivity exceptions that may have been programmed or designed. Hence, we assume that the shapes $\mathbf{X}_{i,j}$, defined each as the intersection of the lattice with tile $\mathbf{T}_{i,j}$ are all similar and form a *SQUINT* pattern. So, we make the list of the elements (disks and balls) that interfere with tile $\mathbf{T}_{i,j}$. We store their relative indices: for example, if disk $\mathbf{D}$[i,j+1,d] interferes with $\mathbf{T}_{i,j}$, we store ¡0,1,d¿. Then, we simply use the solution provided in Section 4.2 to invert $\mathbf{q}$ and to compute, in constant time, the index pair [i,j] of the tile $\mathbf{T}_{i,j}$ that contains $\mathbf{q}$. Then, we use the stored relative indices to identify the disks and bars that interfere with $\mathbf{T}_{i,j}$, instantiate them, and classify $\mathbf{q}$ against each one of them. Typically, the number of such elements is independent of n and small. Hence, the solution has constant time-cost. One may also consider storing the preimage of $\mathbf{L}$, which is a regular lattice (Fig. 24) and classifying the preimage of q against it. Unfortunately, as shown in Fig. 24-b, the inversion warps the disks and bars, which may make PMC against them more challenging, unless a pixelized look-up image (texture) of the generic cell is used to provide approximate, but highly efficient PMC.

5.5 Exceptions in Lattices

Both the total area computation and the PMC approach proposed above need to be adjusted when exceptions are necessary. We define exceptions by a Boolean function for each disk pattern and for each bar pattern. For example, for a bar pattern, that function takes as input the indices of the starting and ending disks of the bar, the index offsets between them, and possibly other parameters. For each bar element of that bar pattern, it defines whether the element is part of the lattice. Exceptions may be used to: (1) Trim dangling elements along the border or add elements where needed to complete a border loop (Fig. 1-a), (2) Remove elements randomly to simulate defects or irregularities, (3) Remove small two-patterns of elements to create a multi-level lattice (Fig. 1-b), and (4) Trim elements of a fine lattice when they do not fit inside a coarse *SQUINT* lattice, hence creating a ***Lattice-in-Lattice*** (***LiL***) model (Fig. 1-c). Here, the LiL exceptions are simply defined by the result of PMC calls that classify disk centers of the finer lattice against the coarse lattice. But more sophisticated criteria may be used. The details, the GPU implementation, and performance of constant cost algorithms for computing the area of LiL's and for performing PMC against them will be addressed in a separate report.

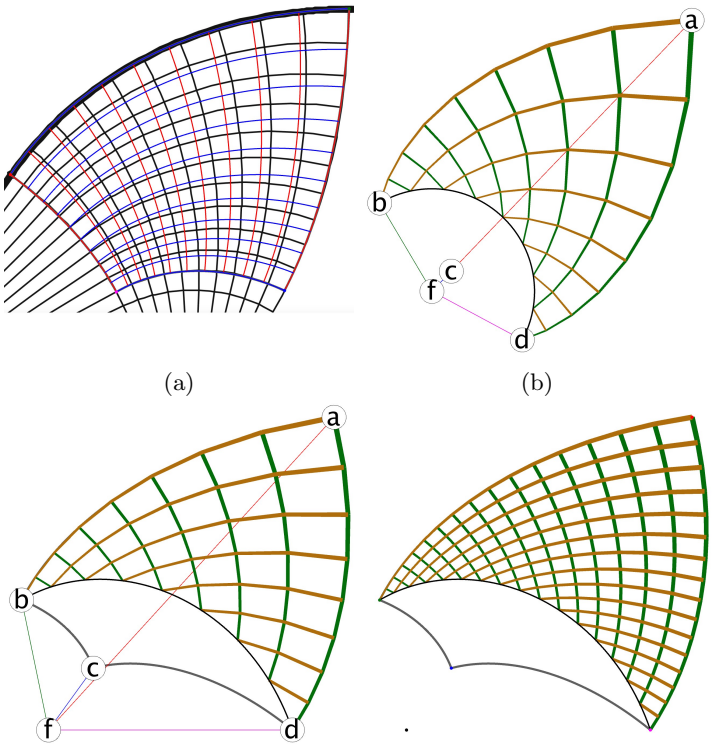


(a) (b)

Figure 24: A *SQUINT* lattice (a) and its preimage (b). Notice that the shapes defined as intersections of the preimage of the lattice with each non-border tile form a regular two-pattern. When considering the border tiles, there are a total of 9 different such shapes.

5.6 Possible relevance to Michell Trusses

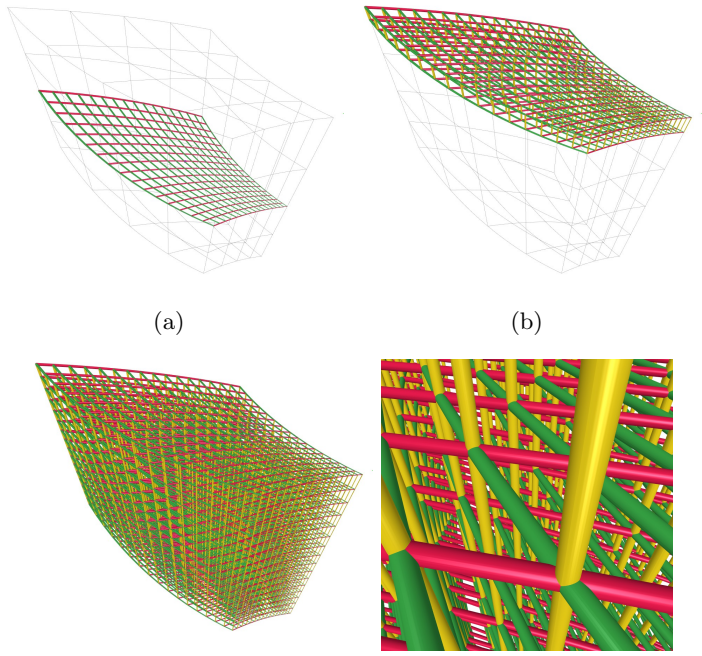
Michell truss-lattices [12] provide a maximum stiffness solution for a cantilever of a given weight carrying a point-load at its free end, $\mathbf{e}$. Under some assumptions, these optimal solutions [6] consist of two families $\{\mathbf{F}_i\}$ and $\{\mathbf{G}_j\}$ of log-spirals with the following properties: The two border-spirals, $\mathbf{F}_0$ and $\mathbf{G}_0$, meet at $\mathbf{e}$. All $\mathbf{F}_i$ meet $\mathbf{G}_0$ orthogonally. All $\mathbf{G}_j$ meet $\mathbf{F}_0$ orthogonally. Although portions of such a Michell lattice may look strikingly similar to a *SQUINT* lattice with the same corner points, they are, in general, different (Fig. 25-a) and not steady. However, optimal Michell truss-lattices that connect a fixed circle to a weight carrying a point-load [21] are a special case of a symmetric and conformal *SQUINT* lattice (Fig. 25-b), suggesting that steadiness may play an important role in lattice optimization and that *SQUINT*-based extensions of these to more general, non-symmetric configurations, when $\lambda_u \neq \lambda_v$ and hence the fixed points are not co-circular (Fig. 25-c and d), deserve further investigation.



(a) (b)

(c) (d)

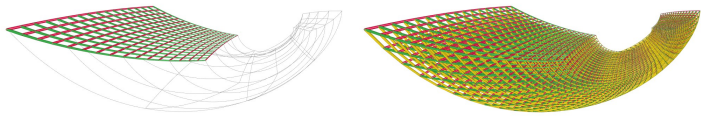
Figure 25: An 11×10 tiles Michell lattice (black) and a 10×10 tiles *SQUINT* lattice (red and blue) with control corners at the corners of a 10×10 tile subset of the Michell lattice (a). A conformal and symmetric *SQUINT* lattice (b) trimmed by a circle with center **f** and radius |**fb**|=|**fd**| corresponds to a version of the Michell lattice that connects that circle to point-load **a**. An asymmetric version (c) for which the circle transforms into a log-spiral, shown refined (d), might prove a useful extension.



(a) (b)

(c) (d)

Figure 26: A single slab that is a 3D "solidification" of a *SQUINT* lattice is shown (a) in the context of the rest of a brick lattice. The first two *SQUINT* lattice slabs of that brick lattice and bars that connect them are shown (b). The whole brick lattice is shown (c) along with a view from the inside (d).



(a) (b)

Figure 27: A slab $\mathbf{L}_0$ (a) and a brick lattice (b) defined by a steady three-pattern that interpolates $\mathbf{L}_0$ and $\mathbf{L}_{n-1}$ and adds inter-slab bars.

5.7 Potential applications for 3D lattices

We hope to explore several three-dimensional extensions of the results presented above. The simplest extension is to turn ("solidify") a planar *SQUINT* lattice into a three-dimensional *SQUINT* slab as follows: (1) Each disk of the *SQUINT* lattice becomes a ball of the same center and radius, and (2) Each bar becomes a cone defined as the revolute sweep of the 2D bar around its axis. An example is shown in Fig. 26-a. For this simple slab-extension, the computational benefits of the *SQUINT* lattice extend easily, both for PMC and for total volume queries. A brick lattice can be made from a steady one-pattern of such *SQUINT* lattice slabs and of inter-slab bars that connect them (Fig. 26-b, -c, and -d). The three-dimensional similarity $\mathcal{W}$ that maps such an initial 3D *SQUINT* slab, $\mathbf{L}_0$ (Fig. 26-b) onto a similar final slab, may be computed easily [4] and its steady morph $\mathcal{W}^w$ may be used to produce any desired number of intermediate slabs, hence defining a steady three-pattern of balls and bars. Adding inter-slab bars produces a brick (In Fig. 26-c). A different configuration is shown in Fig. 27.

The extension of fast PMC and direct total volume queries to such bricks and to more general bricks (with groups of balls that are not necessarily all in the same plane) pose several interesting challenges (steadiness, control points, constraints under which these queries have constant cost), which we hope will be addressed by future research.

6 Summary and conclusion

We define the *SQUINT* map as the composition of two similarity motions that each morph between the opposite edges of a planar quad. We prove that they both have the same fixed-point and that their composition is commutative. We prove that the *SQUINT* map is tranSimilar and provide simple closed-form expressions for computing the image of a point and

its inverse. We suggest possible applications to texture mapping and field display/animation. We also propose a restricted, pseudo-conformal version for which all isocurves cross at right angles and which, when symmetric, is conformal. We show how the *SQUINT* pattern may be used to produce a parameterized procedural model of a *SQUINT* lattice that may be decomposed into a steady two-pattern of quasi-disjoint hubs. We show that its total area can be computed in constant time. We also show that a point may be tested for containment in the *SQUINT* lattice in constant time. Finally, we suggest further work to explore the relation between *SQUINT* and Michell lattices and its extension to three-lattices.

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

The author thanks Ashish Gupta and Kelsey Kurzeja for their comments and Dr. Alex Diaz for pointing out the similarity between steady lattices and Michell trusses.

References

[1] C. Braun, M. Gruendl, C. Marberger, and C. Scherber. 2001. *Beautycheck - Ursachen und Folgen von Attraktivitaet Ursachen und Folgen von Attraktivitaet.* Technical Report. niversity of Regensburg, Germany, Atlanta, GA, USA. https://www.uni-regensburg.de/Fakultaeten/phil_Fak_II/Psychologie/Psy_II/beautycheck/english/index.htm

[2] Renjie Chen and Ofir Weber. 2015. Bounded Distortion Harmonic Mappings in the Plane. *ACM Trans. Graph.* 34, 4, Article 73 (july 2015), 12 pages. https://doi.org/10.1145/2766989

[3] Chen Chu, Greg Graf, and David W Rosen. 2008. Design for additive manufacturing of cellular structures. *Computer-Aided Design and Applications* 5, 5 (2008), 686–696.

[4] Ashish Gupta, George Allen, Suraj Musuvathy, Pranav Kumar, Jarek Rossignac, and Kelsey Kurzeja. 2018. *LM: Designing and processing parametric models of steady lattices.* Technical Report. GVU-2018-02, Georgia Institute of Technology, Atlanta, GA, USA. http://hdl.handle.net/1853/60058

[5] Christoph M. Hoffmann. 1989. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[6] Arvind Sivarama Krishnan. 2015. *Using Michell Truss Principles to find an Optimal Structure Suitable for Additive Manufacturing.* Master's thesis. North Carolina State University. http://www.lib.ncsu.edu/resolver/1840.16/10383

[7] Kelsey Kurzeja and Jarek Rossignac. 2018. *RF: RangeFinder: Accelerating ball-interference queries against steady lattices.* Technical Report. GVU-2018-03, Georgia Institute of Technology, Atlanta, GA, USA. http://hdl.handle.net/1853/60057

[8] F. C. Langbein, C. H. Gao, B. I. Mills, A. D. Marshall, and R. R. Martin. 2004. Topological and Geometric Beautification of Reverse Engineered Geometric Models. In *Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications (SM '04)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 255–260. http://dl.acm.org.prx.library.gatech.edu/citation.cfm?id=1217875.1217915

[9] Shenglan Liu, Ralph R. Martin, Frank C. Langbein, and Paul L. Rosin. 2007. Segmenting Periodic Reliefs on Triangle Meshes. In *Proceedings of the 12th IMA International Conference on Mathematics of Surfaces XII*. Springer-Verlag, Berlin, Heidelberg, 290–306. http://dl.acm.org.prx.library.gatech.edu/citation.cfm?id=1770873.1770891

[10] Xingchen Liu and Vadim Shapiro. 2018. Multiscale shape-material modeling by composition. *Computer-Aided Design* 102 (2018), 194–203.

[11] Yanxi Liu, Robert T. Collins, and Yanghai Tsin. 2004. A Computational Model for Periodic Pattern Perception Based on Frieze and Wallpaper Groups. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 3 (March 2004), 354–371. https://doi.org/10.1109/TPAMI.2004.1262332

[12] Anthony G. M. Michell. 1904. The limit of Economy of Material in Frame-structures. *Phil. Mag., Ser* VI 8 (1904), 589–597.

[13] Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. 2013. Symmetry in 3D Geometry: Extraction and Applications. *Comput. Graph. Forum* 32, 6 (Sept. 2013), 1–23. https://doi.org/10.1111/cgf.12010

[14] Michael Mortenson. 2007. *Geometric Transformations For 3D Modeling* (2nd ed.). Industrial Press, Inc., New York, NY, USA.

[15] Alexander A. Pasko, Oleg Fryazinov, Turlif Vilbrandt, Pierre-Alain Fayolle, and Valery Adzhiev. 2011. Procedural function-based modelling of volumetric microstructures. *Graphical Models* 73 (2011), 165–181.

[16] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J. Guibas. 2008. Discovering Structural Regularity in 3D Geometry. *ACM Trans. Graph.* 27, 3, Article 43 (Aug. 2008), 11 pages. https://doi.org/10.1145/1360612.1360642

[17] William Regli, Jarek Rossignac, Vadim Shapiro, and Vijay Srinivasan. 2016. The new frontiers in computational modeling of material structures. *Computer-Aided Design* 77 (2016), 73–85.

[18] Jarek Rossignac and Álvar Vinacua. 2011. Steady affine motions and morphs. *ACM Transactions on Graphics (TOG)* 30, 5 (2011), 116.

[19] Tobias A Schaedler and William B Carter. 2016. Architected cellular materials. *Annual Review of Materials Research* 46 (2016), 187–210.

[20] Ken Shoemake. 1985. Animating Rotation with Quaternion Curves. *SIGGRAPH Comput. Graph.* 19, 3 (July 1985), 245–254. https://doi.org/10.1145/325165.325242

[21] Robert E. Skelton and Maurcio C. de Oliveira. 2010. Optimal tensegrity structures in bending: The discrete Michell truss. *Journal of the Franklin Institute* 347, 1 (2010), 257 – 283. https://doi.org/10.1016/j.jfranklin.2009.10.009 Dynamics and Control.

[22] Hongqing Wang and David W Rosen. 2002. Parametric modeling method for truss structures. In *ASME Computers and Information in Engineering Conference*.

[23] Matthew Thomas Wojciechowski et al. 2016. Determining Optimal Geometries of Plane Stress Truss Structures using Numerically Mapped Principal Stress Trajectories. (2016).